

C10


```

pgma_restore/restore.c 1
  RSTPL_Finish.....3
  RSTPL_Initialize.....9
  validate_plugin 13
restore_dapi/plugin.cc 17
  RSTPL_ClearRestoreContext 82
  RSTPL_DoesAlreadyExist...89
  RSTPL_Push.....88
  RSTPL_GetAllBackupTime...54
  RSTPL_GetCurrentBackupTime 57
  RSTPL_GetNextLevelObjects...37
  RSTPL_GetTopLevelObjects 29
  RSTPL_GetTopLevelTemplates...79
  RSTPL_Identity 87
  RSTPL_Initialize.....44
  RSTPL_IsAlreadyBackable 89
  RSTPL_IsAlreadyBacked...78
  RSTPL_IsAlreadyBacked...77
  RSTPL_IsThereNextBackupForTime 85
  RSTPL_IsThereNextBackupForTime 85
  RSTPL_IsTherePrevBackupForTime 83
  RSTPL_MarkObject.....44
  RSTPL_SetBackupForTime 59
  RSTPL_SetBackupForTime 59
  RSTPL_SetNextBackup...68
  RSTPL_SetNextBackup...65
  RSTPL_SetPrevBackup...62
  RSTPL_SetTopLevelObject...33
  RSTPL_Submit 27
  RSTPL_UntagMarkObject...47
  dump_time_list 19
  dump_time_list 20
  dump_time_list 18
  dump_time_list.....22
  dump_time_list.....22

```



```

2  /*****
3  **
4  ** File Name:   RSInitfin.c
5  **
6  ** Copyright (c) 1998, 1999 by EMC Corporation.
7  **
8  ** Purpose:
9  **
10 ** This module contains the Restore Service Library
11 ** functions to
12 ** initialize and terminate the restore operation.
13 **
14 ** -----
15 ** RSISL_Initialize
16 ** RSISL_Finish
17 **
18 ** Internal Functions:
19 **
20 **
21 ** Compile-Time Options:
22 ** This section must list any compile time definitions
23 ** which will affect this header.
24 **
25 **
26 **
27 **
28 **
29 **
30 **
31 ** The following provides an RCS id in the binary that can be located
32 ** with the what(1) utility. The intent is to keep this short.
33 **
34 **
35 **
36 **
37 **
38 **
39 **
40 **
41 **
42 **
43 **
44 **
45 **
46 **
47 **
48 **
49 **
50 **
51 **
52 **
53 **
54 **
55 **
56 **
57 **
58 **
59 **
60 **
61 **
62 **
63 **
64 **
65 **
66 **
67 **
68 **
69 **
70 **
71 **
72 **
73 **
74 **
75 **
76 **
77 **
78 **
79 **
80 **
81 **
82 **
83 **
84 **
85 **
86 **
87 **
88 **
89 **
90 **
91 **
92 **
93 **
94 **
95 **
96 **
97 **
98 **
99 **
100 **
101 **
102 **
103 **
104 **
105 **
106 **
107 **
108 **
109 **
110 **
111 **
112 **
113 **
114 **
115 **
116 **
117 **
118 **
119 **
120 **
121 **
122 **
123 **
124 **
125 **
126 **
127 **
128 **
129 **
130 **
131 **
132 **
133 **
134 **
135 **
136 **
137 **
138 **
139 **
140 **
141 **
142 **
143 **
144 **
145 **
146 **
147 **
148 **
149 **
150 **
151 **
152 **
153 **
154 **
155 **
156 **
157 **
158 **
159 **
160 **
161 **
162 **
163 **
164 **
165 **
166 **
167 **
168 **
169 **
170 **
171 **
172 **
173 **
174 **
175 **
176 **
177 **
178 **
179 **
180 **
181 **
182 **
183 **
184 **
185 **
186 **
187 **
188 **
189 **
190 **
191 **
192 **
193 **
194 **
195 **
196 **
197 **
198 **
199 **
200 **
201 **
202 **
203 **
204 **
205 **
206 **
207 **
208 **
209 **
210 **
211 **
212 **
213 **
214 **
215 **
216 **
217 **
218 **
219 **
220 **
221 **
222 **
223 **
224 **
225 **
226 **
227 **
228 **
229 **
230 **
231 **
232 **
233 **
234 **
235 **
236 **
237 **
238 **
239 **
240 **
241 **
242 **
243 **
244 **
245 **
246 **
247 **
248 **
249 **
250 **
251 **
252 **
253 **
254 **
255 **
256 **
257 **
258 **
259 **
260 **
261 **
262 **
263 **
264 **
265 **
266 **
267 **
268 **
269 **
270 **
271 **
272 **
273 **
274 **
275 **
276 **
277 **
278 **
279 **
280 **
281 **
282 **
283 **
284 **
285 **
286 **
287 **
288 **
289 **
290 **
291 **
292 **
293 **
294 **
295 **
296 **
297 **
298 **
299 **
300 **
301 **
302 **
303 **
304 **
305 **
306 **
307 **
308 **
309 **
310 **
311 **
312 **
313 **
314 **
315 **
316 **
317 **
318 **
319 **
320 **
321 **
322 **
323 **
324 **
325 **
326 **
327 **
328 **
329 **
330 **
331 **
332 **
333 **
334 **
335 **
336 **
337 **
338 **
339 **
340 **
341 **
342 **
343 **
344 **
345 **
346 **
347 **
348 **
349 **
350 **
351 **
352 **
353 **
354 **
355 **
356 **
357 **
358 **
359 **
360 **
361 **
362 **
363 **
364 **
365 **
366 **
367 **
368 **
369 **
370 **
371 **
372 **
373 **
374 **
375 **
376 **
377 **
378 **
379 **
380 **
381 **
382 **
383 **
384 **
385 **
386 **
387 **
388 **
389 **
390 **
391 **
392 **
393 **
394 **
395 **
396 **
397 **
398 **
399 **
400 **
401 **
402 **
403 **
404 **
405 **
406 **
407 **
408 **
409 **
410 **
411 **
412 **
413 **
414 **
415 **
416 **
417 **
418 **
419 **
420 **
421 **
422 **
423 **
424 **
425 **
426 **
427 **
428 **
429 **
430 **
431 **
432 **
433 **
434 **
435 **
436 **
437 **
438 **
439 **
440 **
441 **
442 **
443 **
444 **
445 **
446 **
447 **
448 **
449 **
450 **
451 **
452 **
453 **
454 **
455 **
456 **
457 **
458 **
459 **
460 **
461 **
462 **
463 **
464 **
465 **
466 **
467 **
468 **
469 **
470 **
471 **
472 **
473 **
474 **
475 **
476 **
477 **
478 **
479 **
480 **
481 **
482 **
483 **
484 **
485 **
486 **
487 **
488 **
489 **
490 **
491 **
492 **
493 **
494 **
495 **
496 **
497 **
498 **
499 **
500 **
501 **
502 **
503 **
504 **
505 **
506 **
507 **
508 **
509 **
510 **
511 **
512 **
513 **
514 **
515 **
516 **
517 **
518 **
519 **
520 **
521 **
522 **
523 **
524 **
525 **
526 **
527 **
528 **
529 **
530 **
531 **
532 **
533 **
534 **
535 **
536 **
537 **
538 **
539 **
540 **
541 **
542 **
543 **
544 **
545 **
546 **
547 **
548 **
549 **
550 **
551 **
552 **
553 **
554 **
555 **
556 **
557 **
558 **
559 **
560 **
561 **
562 **
563 **
564 **
565 **
566 **
567 **
568 **
569 **
570 **
571 **
572 **
573 **
574 **
575 **
576 **
577 **
578 **
579 **
580 **
581 **
582 **
583 **
584 **
585 **
586 **
587 **
588 **
589 **
590 **
591 **
592 **
593 **
594 **
595 **
596 **
597 **
598 **
599 **
600 **
601 **
602 **
603 **
604 **
605 **
606 **
607 **
608 **
609 **
610 **
611 **
612 **
613 **
614 **
615 **
616 **
617 **
618 **
619 **
620 **
621 **
622 **
623 **
624 **
625 **
626 **
627 **
628 **
629 **
630 **
631 **
632 **
633 **
634 **
635 **
636 **
637 **
638 **
639 **
640 **
641 **
642 **
643 **
644 **
645 **
646 **
647 **
648 **
649 **
650 **
651 **
652 **
653 **
654 **
655 **
656 **
657 **
658 **
659 **
660 **
661 **
662 **
663 **
664 **
665 **
666 **
667 **
668 **
669 **
670 **
671 **
672 **
673 **
674 **
675 **
676 **
677 **
678 **
679 **
680 **
681 **
682 **
683 **
684 **
685 **
686 **
687 **
688 **
689 **
690 **
691 **
692 **
693 **
694 **
695 **
696 **
697 **
698 **
699 **
700 **
701 **
702 **
703 **
704 **
705 **
706 **
707 **
708 **
709 **
710 **
711 **
712 **
713 **
714 **
715 **
716 **
717 **
718 **
719 **
720 **
721 **
722 **
723 **
724 **
725 **
726 **
727 **
728 **
729 **
730 **
731 **
732 **
733 **
734 **
735 **
736 **
737 **
738 **
739 **
740 **
741 **
742 **
743 **
744 **
745 **
746 **
747 **
748 **
749 **
750 **
751 **
752 **
753 **
754 **
755 **
756 **
757 **
758 **
759 **
760 **
761 **
762 **
763 **
764 **
765 **
766 **
767 **
768 **
769 **
770 **
771 **
772 **
773 **
774 **
775 **
776 **
777 **
778 **
779 **
780 **
781 **
782 **
783 **
784 **
785 **
786 **
787 **
788 **
789 **
790 **
791 **
792 **
793 **
794 **
795 **
796 **
797 **
798 **
799 **
800 **
801 **
802 **
803 **
804 **
805 **
806 **
807 **
808 **
809 **
810 **
811 **
812 **
813 **
814 **
815 **
816 **
817 **
818 **
819 **
820 **
821 **
822 **
823 **
824 **
825 **
826 **
827 **
828 **
829 **
830 **
831 **
832 **
833 **
834 **
835 **
836 **
837 **
838 **
839 **
840 **
841 **
842 **
843 **
844 **
845 **
846 **
847 **
848 **
849 **
850 **
851 **
852 **
853 **
854 **
855 **
856 **
857 **
858 **
859 **
860 **
861 **
862 **
863 **
864 **
865 **
866 **
867 **
868 **
869 **
870 **
871 **
872 **
873 **
874 **
875 **
876 **
877 **
878 **
879 **
880 **
881 **
882 **
883 **
884 **
885 **
886 **
887 **
888 **
889 **
890 **
891 **
892 **
893 **
894 **
895 **
896 **
897 **
898 **
899 **
900 **
901 **
902 **
903 **
904 **
905 **
906 **
907 **
908 **
909 **
910 **
911 **
912 **
913 **
914 **
915 **
916 **
917 **
918 **
919 **
920 **
921 **
922 **
923 **
924 **
925 **
926 **
927 **
928 **
929 **
930 **
931 **
932 **
933 **
934 **
935 **
936 **
937 **
938 **
939 **
940 **
941 **
942 **
943 **
944 **
945 **
946 **
947 **
948 **
949 **
950 **
951 **
952 **
953 **
954 **
955 **
956 **
957 **
958 **
959 **
960 **
961 **
962 **
963 **
964 **
965 **
966 **
967 **
968 **
969 **
970 **
971 **
972 **
973 **
974 **
975 **
976 **
977 **
978 **
979 **
980 **
981 **
982 **
983 **
984 **
985 **
986 **
987 **
988 **
989 **
990 **
991 **
992 **
993 **
994 **
995 **
996 **
997 **
998 **
999 **
1000 **

```

```

64  /
65  ** Local headers
66  **
67  #include <RSInitfin.h>
68  #include <RSRestartup.h>
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

127 /*****
128  * RSTSL_initialize:
129  *
130  * This function takes care of all the initialization for a restore
131  * session. This function be called prior to any of the other functions
132  * in the Restore API.
133  *
134  * Parameters:
135  *
136  *   userName (!) - The name of the user.
137  *
138  *****/
139
140 errno_t RSTSL_initialize( const char *userName )
141 {
142     errno_t byStatus = E_SUCCESS;
143
144     /*
145      * If we have not yet allocated space for a restore context
146      * structure, do so now. If we have already done so, just clear it
147      * now.
148      */
149
150     if (NULL == rcp)
151     {
152         rcp = (struct restore_context *)malloc(sizeof(
153             struct restore_context));
154     }
155     if (NULL == rcp)
156     {
157         rec_api_log_error(SUA_CSN_NOMEM, NULL);
158         return (EP_RB_RECOVER_NOMEM);
159     }
160     memset(rcp, 0, sizeof(struct restore_context));
161
162     rcp->rc_human_username = ea_strdup(userName);
163
164     if ((rcp->rc_human_username) {
165         rec_api_log_error(SUA_CSN_NOMEM, NULL);
166         return (EP_RB_RECOVER_NOMEM);
167     }
168
169     /*
170      * Set the appropriate field in the recovery context to indicate
171      * that this recover session is based on the Recover API.
172      * This flag is in place for historical reasons but is used by
173      * other parts of the Recover API library.
174      */
175
176     rcp->rcp_mode = 1;
177
178     /*
179      * Initialize the logging mechanism.
180      */
181
182     if (status = rbrlog_begin(rcp, progname))
183     {
184         return(status);
185     }
186
187     /*
188      * Initialize the few "recover context" variables that we can at
189      * this early stage
190      */
191 }

```

```

191 1
192 1
193 1
194 1
195 1
196 1
197 1
198 1
199 1
200 1
201 1
202 1
203 1
204 1
205 1
206 1
207 1
208 1
209 1
210 1
211 1

```

```

Thu Jan 03 12:52:58 2008                                RSTSL_Finish                                Page 5 of 92

/*****
 * RSTSL_Finish
 *
 * Function Description:
 *
 * This function terminates a restore session,
 * but not while a restore is in
 * progress. It will be rejected if a restore is currently being executed.
 * This routine will clean up any local memory used in the session.
 *
 * Parameters:
 *
 * none
 *
 */
eerrno_t
RSTSL_Finish( void )
{
    int mc_n;

    eerrno_t err = E_SUCCESS;

    if (NULL == rcp)
    {
        return( E_SUCCESS );
    }
    RemoveSubinFiles();
    /* Call rtr_cleanup() which kills the aux proc(s), unlocks the work
     * item, then calls rtrlog_end() to enter the last logs and to close
     * the log file.
     */
    rtr_cleanup(rcp);

    /*
     * Deallocate the memory of restore_context and the related
     * structures.
     */
    if (NULL != rcp->rcp_mcp) /* Free the multiset structures */
    {
        mcp_t_destroy(rcp->rcp_mcp);
    }

    /*
     * Free the mark bit map space
     */
    For (mc_n = 0; mc_n < rcp->rcp_marks.plane_alloc; mc_n++)
    {
        if (NULL != rcp->rcp_marks[mc_n])
        {
            free(rcp->rcp_marks[mc_n]);
        }
        rcp->rcp_marks[mc_n] = NULL;
    }

    if (NULL != rcp->rcp_marks_by_plane)

```

```

Thu Jan 03 12:52:58 2008                                NSTL_Finish                                Page 6 of 92

274 2 {
275 2     free(tcp->rc_marks_by_plane);
276 2 }
277 1
278 1 /*
279 1  * Free the configuration structures
280 1  */
281 1 #if 0
282 1     if (NULL != rcp->rc_cfgname)
283 1     {
284 1         free(rcp->rc_cfgname);
285 1     }
286 1
287 1 #endif
288 1
289 1 if (NULL != rcp->rc_config)
290 1 {
291 2     rcbe_freeconfig(rcp->rc_config);
292 2 }
293 2
294 1 /*
295 1  * Free the OS MOME structures array
296 1  * Note that even though rc_demones is the head of linked list
297 1  * of demons_info structures, the list is allocated via malloc
298 1  * as an array initially (ref. alloc_plane_array()), therefore
299 1  * we can do a free here.
300 1  */
301 1 if (NULL != rcp->rc_demones)
302 1 {
303 2     free(rcp->rc_demones);
304 2 }
305 2
306 1 /*
307 1  * Free the volume list structures.
308 1  */
309 1 if (NULL != rcp->rvlist)
310 1 {
311 2     (void)rvol_validate_destructor(
312 2         rcp->rvlist, RVOL_DESTROY_ALL);
313 2 }
314 1
315 1 /*
316 1  * Free the plugin related data
317 1  */
318 1
319 1 rcp->rc_backup_app = 0;
320 1 while (tcp->currentPipr != rcp->pllist)
321 1 {
322 2     rcp->rc_backup_app++;
323 2     rcp->appdata = rcp->currentPipr->appdata;
324 2     /* allow plugin to clean up and close. so: */
325 2     if ( E_SUCCESS == (ret =
326 2         rcp->currentPipr->plFuncArray[PIFuncIndexFinish] (
327 2             rcp ) ) )
328 2     {
329 3         /* Log error, continue */
330 3         The_user_error(err,
331 3             "ASP2_Finish failed for restore plugin
332 3             library %s\n",
333 3             rcp->currentPipr->idData->name );
334 3     }
335 2 }
336 2
337 2 dlistose( rcp->currentPipr->liblist );
338 2 rcp->pllist = rcp->pllist->next;
339 2 }

```


[illegible]

```

595 1
596 2
597 3
598 4
599 5
600 6
601 7
602 8
603 9
604 10
605 11
606 12
607 13
608 14
609 15
610 16
611 17
612 18
613 19
614 20
615 21
616 22
617 23
618 24
619 25
620 26
621 27
622 28
623 29
624 30
625 31
626 32
627 33
628 34
629 35
630 36
631 37
632 38
633 39
634 40
635 41
636 42
637 43
638 44
639 45
640 46
641 47
642 48
643 49
644 50
645 51
646 52
647 53
648 54
649 55
650 56
651 57
652 58
653 59
654 60
655 61
656 62
657 63
658 64
659 65
660 66
661 67
662 68
663 69
664 70
665 71
666 72
667 73
668 74
669 75
670 76
671 77
672 78
673 79
674 80
675 81
676 82
677 83
678 84
679 85
680 86
681 87
682 88
683 89
684 90
685 91
686 92
687 93
688 94
689 95
690 96
691 97
692 98
693 99
694 100
695 101
696 102
697 103
698 104
699 105
700 106
701 107
702 108
703 109
704 110
705 111
706 112
707 113
708 114
709 115
710 116
711 117
712 118
713 119
714 120
715 121
716 122
717 123
718 124
719 125
720 126
721 127
722 128
723 129
724 130
725 131
726 132
727 133
728 134
729 135
730 136
731 137
732 138
733 139
734 140
735 141
736 142
737 143
738 144
739 145
740 146
741 147
742 148
743 149
744 150
745 151
746 152
747 153
748 154
749 155
750 156
751 157
752 158
753 159
754 160
755 161
756 162
757 163
758 164
759 165
760 166
761 167
762 168
763 169
764 170
765 171
766 172
767 173
768 174
769 175
770 176
771 177
772 178
773 179
774 180
775 181
776 182
777 183
778 184
779 185
780 186
781 187
782 188
783 189
784 190
785 191
786 192
787 193
788 194
789 195
790 196
791 197
792 198
793 199
794 200
795 201
796 202
797 203
798 204
799 205
800 206
801 207
802 208
803 209
804 210
805 211
806 212
807 213
808 214
809 215
810 216
811 217
812 218
813 219
814 220
815 221
816 222
817 223
818 224
819 225
820 226
821 227
822 228
823 229
824 230
825 231
826 232
827 233
828 234
829 235
830 236
831 237
832 238
833 239
834 240
835 241
836 242
837 243
838 244
839 245
840 246
841 247
842 248
843 249
844 250
845 251
846 252
847 253
848 254
849 255
850 256
851 257
852 258
853 259
854 260
855 261
856 262
857 263
858 264
859 265
860 266
861 267
862 268
863 269
864 270
865 271
866 272
867 273
868 274
869 275
870 276
871 277
872 278
873 279
874 280
875 281
876 282
877 283
878 284
879 285
880 286
881 287
882 288
883 289
884 290
885 291
886 292
887 293
888 294
889 295
890 296
891 297
892 298
893 299
894 300
895 301
896 302
897 303
898 304
899 305
900 306
901 307
902 308
903 309
904 310
905 311
906 312
907 313
908 314
909 315
910 316
911 317
912 318
913 319
914 320
915 321
916 322
917 323
918 324
919 325
920 326
921 327
922 328
923 329
924 330
925 331
926 332
927 333
928 334
929 335
930 336
931 337
932 338
933 339
934 340
935 341
936 342
937 343
938 344
939 345
940 346
941 347
942 348
943 349
944 350
945 351
946 352
947 353
948 354
949 355
950 356
951 357
952 358
953 359
954 360
955 361
956 362
957 363
958 364
959 365
960 366
961 367
962 368
963 369
964 370
965 371
966 372
967 373
968 374
969 375
970 376
971 377
972 378
973 379
974 380
975 381
976 382
977 383
978 384
979 385
980 386
981 387
982 388
983 389
984 390
985 391
986 392
987 393
988 394
989 395
990 396
991 397
992 398
993 399
994 400
995 401
996 402
997 403
998 404
999 405
1000 406
1001 407
1002 408
1003 409
1004 410
1005 411
1006 412
1007 413
1008 414
1009 415
1010 416
1011 417
1012 418
1013 419
1014 420
1015 421
1016 422
1017 423
1018 424
1019 425
1020 426
1021 427
1022 428
1023 429
1024 430
1025 431
1026 432
1027 433
1028 434
1029 435
1030 436
1031 437
1032 438
1033 439
1034 440
1035 441
1036 442
1037 443
1038 444
1039 445
1040 446
1041 447
1042 448
1043 449
1044 450
1045 451
1046 452
1047 453
1048 454
1049 455
1050 456
1051 457
1052 458
1053 459
1054 460
1055 461
1056 462
1057 463
1058 464
1059 465
1060 466
1061 467
1062 468
1063 469
1064 470
1065 471
1066 472
1067 473
1068 474
1069 475
1070 476
1071 477
1072 478
1073 479
1074 480
1075 481
1076 482
1077 483
1078 484
1079 485
1080 486
1081 487
1082 488
1083 489
1084 490
1085 491
1086 492
1087 493
1088 494
1089 495
1090 496
1091 497
1092 498
1093 499
1094 500
1095 501
1096 502
1097 503
1098 504
1099 505
1100 506
1101 507
1102 508
1103 50
```

Thu Jan 03 12:52:58 2008	validate_plugin	Page 13 of 92	Thu Jan 03 12:52:58 2008	validate_plugin	Page 14 of 92
626	<pre>/* Init Plugins */</pre>		680 1	<pre>if (idbaptcr->version != RSTPL_VERSION)</pre>	
628	<pre>*****</pre>		681 2	<pre>{ /* only version 1 supported so far */</pre>	
629	<pre>validate_plugin</pre>		682 3	<pre> pldaptcr->idbaptcr = NULL;</pre>	
630			683 2	<pre> return -2;</pre>	
631	<pre> Function Description:</pre>		684 1		
632	<pre> * This function retrieves the addresses of the mandatory plugin</pre>		686 1	<pre> if (idbaptcr->num_types && idbaptcr->val_types)</pre>	
633	<pre> * and stores them in the function pointer array. If any function is missing</pre>		687 2	<pre> /* count cant be positive with null pointer */</pre>	
634	<pre> * it returns -1.</pre>		688 2	<pre> pldaptcr->idbaptcr = NULL;</pre>	
635	<pre> * If then calls the identify function and verifies wthe plugin</pre>		689 1	<pre> return -3;</pre>	
636	<pre> * version, and finds its optional functions. Specific error values are</pre>		692 1	<pre>/* If startRestore option set, get its addr or bust */</pre>	
637	<pre> * returned on version mismatch and missing optional functions.</pre>		693 1	<pre>if ((RSTPL_OPTION_SPECIAL_START</pre>	
638	<pre> * Parameters:</pre>		694 1	<pre> && (NULL == (</pre>	
639	<pre> * Inputs:</pre>		695 1	<pre> pldaptcr->pFuncArray[pFuncIndexStartRestore]</pre>	
640	<pre> pldaptcr {</pre>		696 1	<pre> = (pFuncPtr) displn(pldaptcr->libbapt,</pre>	
641	<pre> I) - pointer to plugin data structure with libbapt set</pre>		697 1	<pre> pFuncNames[pFuncIndexStartRestore])))</pre>	
642	<pre> * Outputs:</pre>		698 1	<pre> } /* OR if special find option set, get its addr or bust */</pre>	
643	<pre> pFuncArray in pldaptcr is loaded with pointers to plugin</pre>		699 1	<pre> ((RSTPL_OPTION_SPECIAL_FIND</pre>	
644	<pre> functions</pre>		700 1	<pre> == (idbaptcr->options & RSTPL_OPTION_MASK_FIND))</pre>	
645	<pre> * Returns:</pre>		701 1	<pre> && ((NULL == (pldaptcr->pFuncArray[pFuncIndexFind]</pre>	
646	<pre> 0 on success</pre>		702 1	<pre> = (pFuncPtr) displn(pldaptcr->libbapt,</pre>	
647	<pre> -1 on any missing required functions</pre>		703 1	<pre> pFuncNames[pFuncIndexFind])))</pre>	
648	<pre> -2 if version validation fails OR identify returns junk</pre>		704 1		
649	<pre> -3 if workitem type validation fails</pre>		705 1	<pre> (NULL == (</pre>	
650	<pre> -4 on any missing optional functions indicated by options</pre>		706 1	<pre> pldaptcr->pFuncArray[pFuncIndexFindBeaulte]</pre>	
651	<pre> flags</pre>		707 1	<pre> = (pFuncPtr) displn(pldaptcr->libbapt,</pre>	
652	<pre> * RL_RR_RECOVER_XXX) for error codes returned from Identify function</pre>		708 1	<pre> pFuncNames[pFuncIndexFindBeaulte])</pre>	
653	<pre> *****</pre>		709 1	<pre> ((RSTPL_OPTION_GETMEDIA option set, get its addr or bust */</pre>	
654	<pre> *****</pre>		710 1	<pre> ((RSTPL_OPTION_SPECIAL_GET_MEDIA</pre>	
655	<pre> *****</pre>		711 1	<pre> == (</pre>	
656	<pre>static int validate_plugin() struct pluginData *pIdbaptcr {</pre>		712 1	<pre> idbaptcr->options & RSTPL_OPTION_MASK_GET_MEDIA))</pre>	
657	<pre>{</pre>		713 1	<pre> && (NULL == (pldaptcr->pFuncArray[pFuncIndexGetMedia]</pre>	
658 1	<pre> index;</pre>		714 1	<pre> = (pFuncPtr) displn(pldaptcr->libbapt,</pre>	
659 1	<pre> struct pluginData *idbaptcr;</pre>		715 1	<pre> pFuncNames[pFuncIndexGetMedia])))</pre>	
660 1	<pre> for(index = 0; index <= pFuncIndexLastBasic; index++)</pre>		716 2	<pre> {</pre>	
661 1	<pre> {</pre>		717 2	<pre> pldaptcr->idbaptcr = NULL;</pre>	
662 1	<pre> if (NULL == (pIdbaptcr->pFuncArray[index]</pre>		718 2	<pre> return -4;</pre>	
663 2	<pre> = (pFuncPtr) displn(pldaptcr->libbapt,</pre>		719 1	<pre> }</pre>	
664 2	<pre> pFuncNames[index]</pre>		720 1	<pre> return 0;</pre>	
665 2	<pre> })</pre>		721 1		
666 2	<pre> return -1;</pre>		722 1		
667 1	<pre> }</pre>				
668 1	<pre> }</pre>				
669 1	<pre> /* call identify and validate */</pre>				
670 1	<pre> status = pldaptcr->pFuncArray[pFuncIndexIdentify] (</pre>				
671 1	<pre> &pIdbaptcr->idbaptcr);</pre>				
672 1	<pre> if (status != E_SUCCESS)</pre>				
673 1	<pre> return status;</pre>				
674 1	<pre> if (NULL == (pIdbaptcr = (</pre>				
675 1	<pre> struct pluginData *)pIdbaptcr->idbaptcr)</pre>				
676 1	<pre> return -2;</pre>				
677 1	<pre> return -2;</pre>				

723

/*

validate_plugin */

```

1 #define _POSIX_SOURCE 1
2
3 #include <restore/restoretree.h>
4 #include <restore/rsidlogin.h>
5 #include <abconfig/rbconfig.h>
6
7 //
8

```

```

10 extern "C" void dump_unix_time(NSTRPC_time_t time)
11 {
12     DateTime t(time);
13     if(t.time)
14     {
15         cout << "Time: " << t << endl;
16     }
17     else
18     {
19         cout << "Time: Zero value\n";
20     }
21 }

```

```

23      extern "C" void dump_time_list(RSTRPC_time_list *list, ostream &out)
24      {
25          DateTime *t;
26          out << "Dump of time list:\n";
27          for(RSTRPC_time_list *listelem=list; listelem; listelem=listelem->next)
28              {
29                  t=new DateTime(listelem->time);
30                  out << "Time: " << *t << endl;
31                  free(t);
32              }
33          out << "End of time list:\n";
34      }
35
36
37

```

```

39      extern "C" void dump_obj_list(RSTRPC_obj_list *list, ostream &out)
40      {
41          out << "Dump of RSTRPC_obj_list:\n";
42          for(list=list->next)
43              {
44                  RSTRPC_obj_level_obj *tobj=list->tobj;
45                  RSTRPC_restorable_obj_root *ro=(tobj->root);
46                  out << "tobj << endl;
47                  out << "level: " << ro->objlevel << " Backup App:" <<
48                      if(ro->objName)
49                          {
50                              {
51                                  out << "
52                                  name : " << ro->objName << endl;
53                                  if(ro->objTypeString)
54                                      {
55                                          {
56                                              out << "
57                                              types: " << ro->objTypeString << endl;
58                                          }
59                                  }
60                                  out << "End of dump RSTRPC_obj_list:\n";
61                              }
62                          }
63          }
64      }
65      return;
66
67
68

```

62

//

```
63 extern "C" void dump_ufo_list(RSTRPC_ufo_list *list, ostream &out)
64 {
65     out << "Dump of RSTRPC_ufo_list\n";
66     for(!list;list->list->next)
67     {
68         out << "We have a node\n";
69     }
70     out << "End of dump RSTRPC_ufo_list\n";
71 }
```


73

//

74

```

    return E_OK;
}

```

75

{

```

    // See if an initial sleep is needed
    /* char *ai_getenv("RSTPL_INITIALIZE_SLEEP");
    if(!ai)
    {
        inc ai=atoi(ai);
        rbe_log_stats(0, "RSTPL_Initialize, sleep of %d seconds", ai);
        sleep(ai);
    }
    else
    {
        rbe_log_stats(0, "RSTPL_Initialize, no delay on startup");
    }
    // Allocate restore context data
    context->appdata=(void *)malloc(sizeof(struct restoreContextData));
    if(!context->appdata)
    {
        rbe_log_stats(0, "plugin.cc - malloc failure");
        return E_RR_RECOVER_MALLOC_FAILURE;
    }
    struct restoreContextData *rcd=
    (struct restoreContextData *) (context->appdata);
    rcd->currentWisetListNode=NULL;
    rcd->currentWisetNode=NULL;
    rcd->currentBackupNode=NULL;
    return E_SUCCESS;
}

```

76

}

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

108

//

110

//

Page 27 of 92	RSTPL_Submit	Thu Jan 03 12:52:58 2008
111	/*	
112	* Submit	
113	* This function creates a submit object from the currently marked	
114	* restorable objects. The ID of the created submit object is passed to	
115	* ERMST_Start to begin execution of the restore.	
116	* Parameters:	
117	* context (I) - Pointer to the restore context	
118	* hostname (I) - host to restore to (only if inplace == False)	
119	* policy (I) - The overwrite policy to use	
120	* inplace (I) - Flag if the restore is to be in original locations	
121	* directory (I) - directory to restore to (only if inplace == False)	
122	* transport (I) - Indicator of transport the restoration is to be over (SCSI	
123	* or network)	
124	* submitObjIDptr (I) - ID of the submit user object created to describe	
125	* the restore	
126	* progressCB (I) - pointer to callback function to report progress and	
127	* test for cancellation	
128	* Note: The Progress Callback is currently not used. If the need	
129	* for this developer, the routines which use this argument	
130	* (mark, unmark, submit) must be enhanced.	
131	*****	
132	*****	
133	*****	
134	*****	
135	*****	
136	*****	
137	*****	
138	*****	
139	*****	
140	*****	
141	*****	
142	*****	
143	*****	
144	*****	
145	*****	
146	*****	
147	*****	
148	*****	
149	*****	
150	*****	
151	*****	
152	*****	
153	*****	
154	*****	
155	*****	
156	*****	
157	*****	
158	*****	
159	*****	
160	*****	
161	*****	
162	*****	
163	*****	
164	*****	
165	*****	
166	*****	
167	*****	
168	*****	
169	*****	
170	*****	
171	*****	
172	*****	
173	*****	
174	*****	
175	*****	
176	*****	
177	*****	
178	*****	
179	*****	
180	*****	
181	*****	
182	*****	
183	*****	
184	*****	
185	*****	
186	*****	
187	*****	
188	*****	
189	*****	
190	*****	
191	*****	
192	*****	
193	*****	
194	*****	
195	*****	
196	*****	
197	*****	
198	*****	
199	*****	
200	*****	
201	*****	
202	*****	
203	*****	
204	*****	
205	*****	
206	*****	
207	*****	
208	*****	
209	*****	
210	*****	
211	*****	
212	*****	
213	*****	
214	*****	
215	*****	
216	*****	
217	*****	
218	*****	
219	*****	
220	*****	
221	*****	
222	*****	
223	*****	
224	*****	
225	*****	
226	*****	
227	*****	
228	*****	
229	*****	
230	*****	
231	*****	
232	*****	
233	*****	
234	*****	
235	*****	
236	*****	
237	*****	
238	*****	
239	*****	
240	*****	
241	*****	
242	*****	
243	*****	
244	*****	
245	*****	
246	*****	
247	*****	
248	*****	
249	*****	
250	*****	
251	*****	
252	*****	
253	*****	
254	*****	
255	*****	
256	*****	
257	*****	
258	*****	
259	*****	
260	*****	
261	*****	
262	*****	
263	*****	
264	*****	
265	*****	
266	*****	
267	*****	
268	*****	
269	*****	
270	*****	
271	*****	
272	*****	
273	*****	
274	*****	
275	*****	
276	*****	
277	*****	
278	*****	
279	*****	
280	*****	
281	*****	
282	*****	
283	*****	
284	*****	
285	*****	
286	*****	
287	*****	
288	*****	
289	*****	
290	*****	
291	*****	
292	*****	
293	*****	
294	*****	
295	*****	
296	*****	
297	*****	
298	*****	
299	*****	
300	*****	
301	*****	
302	*****	
303	*****	
304	*****	
305	*****	
306	*****	
307	*****	
308	*****	
309	*****	
310	*****	
311	*****	
312	*****	
313	*****	
314	*****	
315	*****	
316	*****	
317	*****	
318	*****	
319	*****	
320	*****	
321	*****	
322	*****	
323	*****	
324	*****	
325	*****	
326	*****	
327	*****	
328	*****	
329	*****	
330	*****	
331	*****	
332	*****	
333	*****	
334	*****	
335	*****	
336	*****	
337	*****	
338	*****	
339	*****	
340	*****	
341	*****	
342	*****	
343	*****	
344	*****	
345	*****	
346	*****	
347	*****	
348	*****	
349	*****	
350	*****	
351	*****	
352	*****	
353	*****	
354	*****	
355	*****	
356	*****	
357	*****	
358	*****	
359	*****	
360	*****	
361	*****	
362	*****	
363	*****	
364	*****	
365	*****	
366	*****	
367	*****	
368	*****	
369	*****	
370	*****	
371	*****	
372	*****	
373	*****	
374	*****	
375	*****	
376	*****	
377	*****	
378	*****	
379	*****	
380	*****	
381	*****	
382	*****	
383	*****	
384	*****	
385	*****	
386	*****	
387	*****	
388	*****	
389	*****	
390	*****	
391	*****	
392	*****	
393	*****	
394	*****	
395	*****	
396	*****	
397	*****	
398	*****	
399	*****	
400	*****	
401	*****	
402	*****	
403	*****	
404	*****	
405	*****	
406	*****	
407	*****	
408	*****	
409	*****	
410	*****	
411	*****	
412	*****	
413	*****	
414	*****	
415	*****	
416	*****	
417	*****	
418	*****	
419	*****	
420	*****	
421	*****	
422	*****	
423	*****	
424	*****	
425	*****	
426	*****	
427	*****	
428	*****	
429	*****	
430	*****	
431	*****	
432	*****	
433	*****	
434	*****	
435	*****	
436	*****	
437	*****	
438	*****	
439	*****	
440	*****	
441	*****	
442	*****	
443	*****	
444	*****	
445	*****	
446	*****	
447	*****	
448	*****	
449	*****	
450	*****	
451	*****	
452	*****	
453	*****	
454	*****	
455	*****	
456	*****	
457	*****	
458	*****	
459	*****	
460	*****	
461	*****	
462	*****	
463	*****	
464	*****	
465	*****	
466	*****	
467	*****	
468	*****	
469	*****	
470	*****	
471	*****	
472	*****	
473	*****	
474	*****	
475	*****	
476	*****	
477	*****	
478	*****	
479	*****	
480	*****	
481	*****	
482	*****	
483	*****	
484	*****	
485	*****	
486	*****	
487	*****	
488	*****	
489	*****	
490	*****	
491	*****	
492	*****	
493	*****	
494	*****	
495	*****	
496	*****	
497	*****	
498	*****	
499	*****	
500	*****	
501	*****	
502	*****	
503	*****	
504	*****	
505	*****	
506	*****	
507	*****	
508	*****	
509	*****	
510	*****	
511	*****	
512	*****	
513	*****	
514	*****	
515	*****	
516	*****	
517	*****	
518	*****	
519	*****	
520	*****	
521	*****	
522	*****	
523	*****	
524	*****	
525	*****	
526	*****	
527	*****	
528	*****	
529	*****	
530	*****	
531	*****	
532	*****	
533	*****	
534	*****	
535	*****	
536	*****	
537	*****	
538	*****	
539	*****	
540	*****	
541	*****	
542	*****	
543	*****	
544	*****	
545	*****	
546	*****	
547	*****	
548	*****	
549	*****	
550	*****	
551	*****	
552	*****	
553	*****	
554	*****	
555	*****	
556	*****	
557	*****	
558	*****	
559	*****	
560	*****	
561	*****	
562	*****	
563	*****	
564	*****	
565	*****	
566	*****	
567	*****	
568	*****	
569	*****	
570	*****	
571	*****	
572	*****	
573	*****	
574	*****	
575	*****	
576	*****	
577	*****	
578	*****	
579	*****	
580	*****	
581	*****	
582	*****	
583	*****	
584	*****	
585	*****	
586	*****	
587	*****	
588	*****	
589	*****	
590	*****	
591	*****	
592	*****	
593	*****	
594	*****	
595	*****	
596	*****	
597	*****	
598	*****	
599	*****	
600	*****	
601	*****	
602	*****	
603	*****	
604	*****	
605	*****	
606	*****	
607	*****	
608	*****	
609	*****	
610	*****	
611	*****	
612	*****	
613	*****	
614	*****	
615	*****	
616		

Thu Jan 03 12:52:58 2008	RSTPL_GetTopLevelObjects	Page 29 of 92	Thu Jan 03 12:52:58 2008	RSTPL_GetTopLevelObjects	Page 30 of 92
<pre> 190 //***** 191 * Get Top Level Objects 192 * 193 * This function is called to retrieve the configurable backup 194 * items for network backups and work item sets for Symmetric backup, 195 * which are restorable for the given client. 196 * 197 * It is a GOAL of this routine to return all objects ever backed 198 * up successfully. For network backups, though, it only looks in the config 199 * file for 'top level objects' of the given client. 200 * While the restore API will be called repeatedly to retrieve a 201 * number of items on each call, 202 * this plug-in call must retrieve the whole 203 * set of applicable backup objects. 204 * The generic restore service library 205 * will manage the composite list of top level objects from all 206 * backup apps. 207 * 208 * Parameters: 209 * context (I) - Pointer to the restore context 210 * sourceHost (I) - The name of the host whose backups are being restored 211 * topLevelObjs (O) - ptr to linked list of Top Level Objects 212 * numberEntries (O) - the real number of objects returned in the list 213 * 214 * Returns: 215 * R_SUCCESS on success 216 * RP_RB_RECOVER_XXX on error 217 * 218 ****** 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065 1066 1067 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083 1084 1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096 1097 1098 1099 1100 1101 1102 1103 1104 1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115 1116 1117 1118 1119 1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131 1132 1133 1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146 1147 1148 1149 1150 1151 1152 1153 1154 1155 1156 1157 1158 1159 1160 1161 1162 1163 1164 1165 1166 1167 1168 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181 1182 1183 1184 1185 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197 1198 1199 1200 1201 1202 1203 1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301 1302 1303 1304 1305 1306 1307 1308 1309 1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328 1329 1330 1331 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341 1342 1343 1344 1345 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 1358 1359 1360 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392 1393 1394 1395 1396 1397 1398 1399 1400 1401 1402 1403 1404 1405 1406 1407 1408 1409 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425 1426 1427 1428 1429 1430 1431 1432 1433 1434 1435 1436 1437 1438 1439 1440 1441 1442 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454 1455 1456 1457 1458 1459 1460 1461 1462 1463 1464 1465 1466 1467 1468 1469 1470 1471 1472 1473 1474 1475 1476 1477 1478 1479 1480 1481 1482 1483 1484 1485 1486 1487 1488 1489 1490 1491 1492 1493 1494 1495 1496 1497 1498 1499 1500 1501 1502 1503 1504 1505 1506 1507 1508 1509 1510 1511 1512 1513 1514 1515 1516 1517 1518 1519 1520 1521 1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540 1541 1542 1543 1544 1545 1546 1547 1548 1549 1550 1551 1552 1553 1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565 1566 1567 1568 1569 1570 1571 1572 1573 1574 1575 1576 1577 1578 1579 1580 1581 1582 1583 1584 1585 1586 1587 1588 1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1610 1611 1612 1613 1614 1615 1616 1617 1618 1619 1620 1621 1622 1623 1624 1625 1626 1627 1628 1629 1630 1631 1632 1633 1634 1635 1636 1637 1638 1639 1640 1641 1642 1643 1644 1645 1646 1647 1648 1649 1650 1651 1652 1653 1654 1655 1656 1657 1658 1659 1660 1661 1662 1663 1664 1665 1666 1667 1668 1669 1670 1671 1672 1673 1674 1675 1676 1677 1678 1679 1680 1681 1682 1683 1684 1685 1686 1687 1688 1689 1690 1691 1692 1693 1694 1695 1696 1697 1698 1699 1700 1701 1702 1703 1704 1705 1706 1707 1708 1709 1710 1711 1712 1713 1714 1715 1716 1717 1718 1719 1720 1721 1722 1723 1724 1725 1726 1727 1728 1729 1730 1731 1732 1733 1734 1735 1736 1737 1738 1739 1740 1741 1742 1743 1744 1745 1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760 1761 1762 1763 1764 1765 1766 1767 1768 1769 1770 1771 1772 1773 1774 1775 1776 1777 1778 1779 1780 1781 1782 1783 1784 1785 1786 1787 1788 1789 1790 1791 1792 1793 1794 1795 1796 1797 1798 1799 1800 1801 1802 1803 1804 1805 1806 1807 1808 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818 1819 1820 1821 1822 1823 1824 1825 1826 1827 1828 1829 1830 1831 1832 1833 1834 1835 1836 1837 1838 1839 1840 1841 1842 1843 1844 1845 1846 1847 1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860 1861 1862 1863 1864 1865 1866 1867 1868 1869 1870 1871 1872 1873 1874 1875 1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889 1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903 1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917 1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931 1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026 2027 2028 2029 2030 2031 2032 2033 2034 2035 2036 2037 2038 2039 2040 2041 2042 2043 2044 2045 2046 2047 2048 2049 2050 2051 2052 2053 2054 2055 2056 2057 2058 2059 2060 2061 2062 2063 2064 2065 2066 2067 2068 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103 2104 2105 2106 2107 2108 2109 2110 2111 2112 2113 2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144 2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158 2159 2160 2161 2162 2163 2164 2165 2166 2167 2168 2169 2170 2171 2172 2173 2174 2175 2176 2177 2178 2179 2180 2181 2182 2183 2184 2185 2186 2187 2188 2189 2190 2191 2192 2193 2194 2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221 2222 2223 2224 2225 2226 2227 2228 2229 2230 2231 2232 2233 2234 2235 2236 2237 2238 2239 2240 2241 2242 2243 2244 2245 2246 2247 2248 2249 2250 2251 2252 2253 2254 2255 2256 2257 2258 2259 2260 2261 2262 2263 2264 2265 2266 2267 2268 2269 2270 2271 2272 2273 2274 2275 2276 2277 2278 2279 2280 2281 2282 2283 2284 2285 2286 2287 2288 2289 2290 2291 2292 2293 2294 2295 2296 2297 2298 2299 2300 2301 2302 2303 2304 2305 2306 2307 2308 2309 2310 2311 2312 2313 2314 2315 2316 2317 2318 2319 2320 2321 2322 2323 2324 2325 2326 2327 2328 2329 2330 2331 2332 2333 2334 2335 2336 2337 2338 2339 2340 2341 2342 2343 2344 2345 2346 2347 2348 2349 2350 2351 2352 2353 2354 2355 2356 2357 2358 2359 2360 2361 2362 2363 2364 2365 2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390 2391 2392 2393 2394 2395 2396 2397 2398 2399 2400 2401 2402 2403 2404 2405 2406 2407 2408 2409 2410 2411 2412 2413 2414 2415 2416 2417 2418 2419 2420 2421 2422 2423 2424 2425 2426 2427 2428 2429 2430 2431 2432 2433 2434 2435 2436 2437 2438 2439 2440 2441 2442 2443 2444 2445 2446 2447 2448 2449 2450 2451 2452 2453 2454 2455 2456 2457 2458 2459 2460 2461 2462 2463 2464 2465 2466 2467 2468 2469 2470 2471 2472 2473 2474 2475 2476 2477 2478 2479 2480 2481 2482 2483 2484 2485 2486 2487 2488 2489 2490 2491 2492 2493 2494 2495 2496 2497 2498 2499 2500 2501 2502 2503 2504 2505 2506 2507 2508 2509 2510 2511 2512 2513 2514 2515 2516 2517 2518 2519 2520 2521 2522 2523 2524 2525 2526 2527 2528 2529 2530 2531 2532 2533 2534 2535 2536 2537 2538 2539 2540 2541 2542 2543 2544 2545 2546 2547 2548 2549 2550 2551 2552 2553 2554 2555 2556 2557 2558 2559 2560 2561 2562 2563 2564 2565 2566 2567 2568 2569 2570 2571 2572 2573 2574 2575 2576 2577 2578 2579 2580 2581 2582 2583 2584 2585 2586 2587 2588 2589 2590 2591 2592 2593 2594 2595 2596 2597 2598 2599 2600 2601 2602 2603 2604 2605 2606 2607 2608 2609 2610 2611 2612 2613 2614 2615 2616 2617 2618 2619 2620 2621 2622 2623 2624 2625 2626 2627 2628 2629 2630 2631 2632 2633 2634 2635 2636 2637 2638 2639 2640 2641 2642 2643 2644 2645 2646 2647 2648 2649 2650 2651 2652 2653 2654 2655 2656 2657 2658 2659 2660 2661 2662 2663 2664 2665 2666 2667 2668 2669 2670 2671 2672 2673 2674 2675 2676 2677 2678 2679 2680 2681 2682 2683 2684 2685 2686 2687 2688 2689 2690 2691 2692 2693 2694 2695 2696 2697 2698 2699 2700 2701 2702 2703 2704 2705 2706 2707 2708 2709 2710 2711 2712 2713 2714 2715 2716 2717 2718 2719 2720 2721 2722 2723 2724 2725 2726 2727 2728 2729 2730 2731 2732 2733 2734 2735 2736 2737 2738 2739 2740 2741 2742 2743 2744 2745 2746 2747 2748 2749 2750 2751 2752 2753 2754</pre>					

Page 31 of 92	HS/PL_GetPolymorphicObjects	Thu Jan 03 12:52:58 2008	Page 32 of 92		Thu Jan 03 12:52:58 2008
305 2	<code>new_top->root_objLevel=STRIPC_tio_type;</code>		350	<code>//</code>	
306 2	<code>new_top->root_objName=esl_strdup((char *) (m->getNameChar()));</code>				
307 2	<code>if(NULL==new_top->root_objName)</code>				
308 3	<code>{</code>				
309 3	<code> rbe_log_stats{</code>				
	<code> 0,"plugin.cc - allocation failure from esl_strdup failure";</code>				
	<code> return EP_RB_RECOVER_MALLOC_FAILURE;</code>				
	<code> }</code>				
310 2	<code>new_top->wBIC=NULL;</code>				
311 2	<code>new_top->templateName=esl_strdup(""); // Blank as placeholder</code>				
312 2	<code>new_top->root_objTypeString=esl_strdup("");</code>				
313 2	<code>new_top->hostname=esl_strdup(rcd->currentHostListNode->hostname)</code>				
314 2	<code>new_top->filePath=esl_strdup(""); // Blank as placeholder</code>				
315 2	<code>if(NULL == new_top->root_objTypeString </code>				
316 2	<code> NULL == new_top->hostname </code>				
317 2	<code> NULL == new_top->filePath)</code>				
318 2	<code>{</code>				
319 2	<code> rbe_log_stats{</code>				
320 2	<code> 0,"plugin.cc - allocation failure from esl_strdup";</code>				
321 2	<code> return EP_RB_RECOVER_MALLOC_FAILURE;</code>				
322 2	<code> }</code>				
323 2	<code>new_top->wBIC=NULL;</code>				
324 2	<code>new_top->asThread=FALSE;</code>				
325 2	<code>new_list_elem->file=new_top;</code>				
326 2	<code>new_list_elem->next=NULL;</code>				
327 2	<code>if(previous)</code>				
328 2	<code>{</code>				
329 2	<code> previous->next=new_list_elem;</code>				
330 2	<code>}</code>				
331 2	<code>else</code>				
332 2	<code>{</code>				
333 2	<code> *topLevelObj=&new_list_elem;</code>				
334 2	<code> previous=new_list_elem;</code>				
335 2	<code>}</code>				
336 2	<code>return E_SUCCESS;</code>				
337 2	<code>}</code>				
338 2	<code>return E_SUCCESS;</code>				
339 2	<code>}</code>				
340 2	<code>}</code>				
341 2	<code>}</code>				
342 2	<code>}</code>				
343 2	<code>}</code>				
344 2	<code>}</code>				
345 1	<code>}</code>				
346 1	<code>}</code>				
347 1	<code>}</code>				
348 1	<code>}</code>				

```

352  /*****
353  * Set TopLevel Object:
354  *****/
355  // This function is called to notify the plug-in that one of its top
356  // level
357  // objects has been selected (for browsing and marking). The plug-in
358  // should perform whatever validation and initialization is necessary
359  // for browsing and marking this top level object.
360  // If this call returns success,
361  // then RSPPL_GetNextLevelObjects will be
362  // called to retrieve the highest level restorable objects for this
363  // backup object.
364  // NOTE: This function is responsible for setting the template name
365  // and
366  // saving selected elements of the restore context.
367  * Returns:
368  *   E_SUCCESS      on success
369  *   EP_RA_RECOVER_xxx on error
370  *
371  * Parameters:
372  *   (1) - Pointer to the restore context
373  *   context
374  *   backupObject (1) - the selected top level object
375  *
376  *
377  *
378  *
379  *
380  *
381  *
382  *
383  *
384  *
385  *
386  *
387  *
388  *
389  *
390  *
391  *
392  *
393  *
394  *
395  *
396  *
397  *
398  *
399  *
400  *
401  *
402  *
403  *
404  *
405  *
406  *
407  *
408  *
409  *
410  *
411  *
412  *
413  *
414  *
415  *
416  *
417  *
418  *
419  *
420  *
421  *
422  *
423  *
424  *
425  *
426  *
427  *
428  *
429  *
430  *
431  *
432  *
433  *
434  *
435  *
436  *
437  *
438  *
439  *
440  *
441  *
442  *
443  *
444  *
445  *
446  *
447  *
448  *
449  *
450  *
451  *
452  *
453  *
454  *
455  *
456  *
457  *
458  *
459  *
460  *
461  *
462  *
463  *
464  *
465  *
466  *
467  *
468  *
469  *
470  *
471  *
472  *
473  *
474  *
475  *
476  *
477  *
478  *
479  *
480  *
481  *
482  *
483  *
484  *
485  *
486  *
487  *
488  *
489  *
490  *
491  *
492  *
493  *
494  *
495  *
496  *
497  *
498  *
499  *
500  *
501  *
502  *
503  *
504  *
505  *
506  *
507  *
508  *
509  *
510  *
511  *
512  *
513  *
514  *
515  *
516  *
517  *
518  *
519  *
520  *
521  *
522  *
523  *
524  *
525  *
526  *
527  *
528  *
529  *
530  *
531  *
532  *
533  *
534  *
535  *
536  *
537  *
538  *
539  *
540  *
541  *
542  *
543  *
544  *
545  *
546  *
547  *
548  *
549  *
550  *
551  *
552  *
553  *
554  *
555  *
556  *
557  *
558  *
559  *
560  *
561  *
562  *
563  *
564  *
565  *
566  *
567  *
568  *
569  *
570  *
571  *
572  *
573  *
574  *
575  *
576  *
577  *
578  *
579  *
580  *
581  *
582  *
583  *
584  *
585  *
586  *
587  *
588  *
589  *
590  *
591  *
592  *
593  *
594  *
595  *
596  *
597  *
598  *
599  *
600  *
601  *
602  *
603  *
604  *
605  *
606  *
607  *
608  *
609  *
610  *
611  *
612  *
613  *
614  *
615  *
616  *
617  *
618  *
619  *
620  *
621  *
622  *
623  *
624  *
625  *
626  *
627  *
628  *
629  *
630  *
631  *
632  *
633  *
634  *
635  *
636  *
637  *
638  *
639  *
640  *
641  *
642  *
643  *
644  *
645  *
646  *
647  *
648  *
649  *
650  *
651  *
652  *
653  *
654  *
655  *
656  *
657  *
658  *
659  *
660  *
661  *
662  *
663  *
664  *
665  *
666  *
667  *
668  *
669  *
670  *
671  *
672  *
673  *
674  *
675  *
676  *
677  *
678  *
679  *
680  *
681  *
682  *
683  *
684  *
685  *
686  *
687  *
688  *
689  *
690  *
691  *
692  *
693  *
694  *
695  *
696  *
697  *
698  *
699  *
700  *
701  *
702  *
703  *
704  *
705  *
706  *
707  *
708  *
709  *
710  *
711  *
712  *
713  *
714  *
715  *
716  *
717  *
718  *
719  *
720  *
721  *
722  *
723  *
724  *
725  *
726  *
727  *
728  *
729  *
730  *
731  *
732  *
733  *
734  *
735  *
736  *
737  *
738  *
739  *
740  *
741  *
742  *
743  *
744  *
745  *
746  *
747  *
748  *
749  *
750  *
751  *
752  *
753  *
754  *
755  *
756  *
757  *
758  *
759  *
760  *
761  *
762  *
763  *
764  *
765  *
766  *
767  *
768  *
769  *
770  *
771  *
772  *
773  *
774  *
775  *
776  *
777  *
778  *
779  *
780  *
781  *
782  *
783  *
784  *
785  *
786  *
787  *
788  *
789  *
790  *
791  *
792  *
793  *
794  *
795  *
796  *
797  *
798  *
799  *
800  *
801  *
802  *
803  *
804  *
805  *
806  *
807  *
808  *
809  *
810  *
811  *
812  *
813  *
814  *
815  *
816  *
817  *
818  *
819  *
820  *
821  *
822  *
823  *
824  *
825  *
826  *
827  *
828  *
829  *
830  *
831  *
832  *
833  *
834  *
835  *
836  *
837  *
838  *
839  *
840  *
841  *
842  *
843  *
844  *
845  *
846  *
847  *
848  *
849  *
850  *
851  *
852  *
853  *
854  *
855  *
856  *
857  *
858  *
859  *
860  *
861  *
862  *
863  *
864  *
865  *
866  *
867  *
868  *
869  *
870  *
871  *
872  *
873  *
874  *
875  *
876  *
877  *
878  *
879  *
880  *
881  *
882  *
883  *
884  *
885  *
886  *
887  *
888  *
889  *
890  *
891  *
892  *
893  *
894  *
895  *
896  *
897  *
898  *
899  *
900  *
901  *
902  *
903  *
904  *
905  *
906  *
907  *
908  *
909  *
910  *
911  *
912  *
913  *
914  *
915  *
916  *
917  *
918  *
919  *
920  *
921  *
922  *
923  *
924  *
925  *
926  *
927  *
928  *
929  *
930  *
931  *
932  *
933  *
934  *
935  *
936  *
937  *
938  *
939  *
940  *
941  *
942  *
943  *
944  *
945  *
946  *
947  *
948  *
949  *
950  *
951  *
952  *
953  *
954  *
955  *
956  *
957  *
958  *
959  *
960  *
961  *
962  *
963  *
964  *
965  *
966  *
967  *
968  *
969  *
970  *
971  *
972  *
973  *
974  *
975  *
976  *
977  *
978  *
979  *
980  *
981  *
982  *
983  *
984  *
985  *
986  *
987  *
988  *
989  *
990  *
991  *
992  *
993  *
994  *
995  *
996  *
997  *
998  *
999  *
1000  */

```

```

412 1      rcd->currentBackNode = (WtSetNode *)nodep;
413 1      //
414 1      // Set the current backup node to the most recent one in time
415 1      // which is FULL
416 1      //
417 1      //
418 1      // Clear looks if they exist
419 1      if(NULL != rcd->currentBackNode)
420 1      {
421 2          rcd->currentBackNode->unLockWorkItems(context);
422 2          rcd->currentBackNode->unmarkNode(TRUE, FALSE);
423 2      }
424 1      rcd->currentBackNode=NULL;
425 1
426 1      BackupNode *nbase=NULL;
427 1      BackupNode *nextNode;
428 1      int more=1;
429 1
430 1      for(BackupNode *bn=(BackupNode *)nodep->getFirstChildId();
431 1          more;
432 1          bn=nextNode)
433 1      {
434 2          nextNode=(BackupNode *) (nodep->getNextChildId());
435 2
436 2          if(NULL == nextNode) // Force end of loop if no more items to
437 2              // scan
438 2              {
439 3                  more=0;
440 2              }
441 2
442 2          //
443 2          // Populate in case this is necessary
444 2          //
445 2          // Errors will set appropriate state bits on nodes
446 2          int err=bn->populate(context, POPULATE_CHILDREN);
447 2
448 2          bbase=bn; // Save in case we have to return an incomplete one
449 2          if(bn->getCreatable()<WtRE_CONFIGURE)
450 2          {
451 3              // If we already have a current backup node, clear all marks
452 3              if(rcd->currentBackNode)
453 3                  clear_all_marks
454 3              {
455 4                  rcd->currentBackNode->unLockWorkItems(context);
456 4                  rcd->currentBackNode->unmarkNode(TRUE, FALSE);
457 4              }
458 3              rcd->currentBackNode=bn;
459 3              if(!rcd->currentBackNode->lockWorkItems(context))
460 3              {
461 4                  return ER_PB_RECOVER_MILLOCKED;
462 3              }
463 3              return ER_PB_RECOVER_MILLOCKED;
464 3          }
465 3          return ER_PB_RECOVER_NO_TEMPLATE_PROPERTY;
466 3      }
467 3      char * templateBase=(char *)(((
468 3          hPropObject *templateBase=bn->getProperty(TEMPLATE);
469 3          context->rc.template_name=ai_strdup(templateBase);
470 3          if(NULL == templateBase)
471 3              if(NULL == backupObject->templateName ||
472 3                  NULL == context->rc.template_name)
473 3              {

```

rescue_dcpdump.c:18

Page 34 of 92

```

474 4         RSTP_SatOpLevelObject
475 4         the_log_stats{
476 3             0, "Plugin.cc - allocation failure from esi_strdup*");
477 3             return RP_RR_RECOVER_MALLOC_FAILURES;
478 3         }
479 3         // Break out so that we get the first one on the list
480 3         // (most recent)
481 3         more=0;
482 2     }
483 1     }
484 1     }
485 1     //
486 1     // Check and make sure we got one
487 1     //
488 1     if(!rcd->currentBackupNode)
489 1     {
490 2         if(NULL!=bnsave)
491 2         {
492 3             // We shouldn't be able to get here,
493 3             // but let's check for safety
494 3             the_log_stats(0, "could not find any backup node");
495 3             return RP_RR_RECOVER_NONE_EXISTS;
496 2         }
497 2         if(NULL!=rcd->currentBackupNode)
498 2         {
499 3             rcd->currentBackupNode->unlockWorkItems(context);
500 2         }
501 2         rcd->currentBackupNode=bnsave;
502 2         if(!rcd->currentBackupNode->lockWorkItems(context))
503 2         {
504 3             return RP_RR_RECOVER_ML_LOCKED;
505 2         }
506 2         RtrProperty *tppnode=rcd->currentBackupNode->getRtrProperty(
507 2             PROPERTY_TEMPLATE);
508 2         if(!tppnode)
509 2         {
510 3             return RP_RR_RECOVER_NO_COMPLETE_BACKUP;
511 2         }
512 2         char * templateBase=(char *)(((
513 2             RtrPropertyChar *)tppnode)->getValue());
514 2         backupObject->templateName=esi_strdup(templateBase);
515 2         context->src_template_name=esi_strdup(templateBase);
516 2         if(NULL!=backupObject->templateName ||
517 2             NULL!=context->src_template_name)
518 2         {
519 3             the_log_stats(
520 3                 0, "Plugin.cc - allocation failure from esi_strdup*");
521 2             return RP_RR_RECOVER_MALLOC_FAILURE;
522 1         }
523 1         return E_SUCCESS;
524 1     }
525 1     }

```

Thu Jan 03 12:52:58 2008	RSTPL_GetParentLevelObjects	Page 37 of 92	Thu Jan 03 12:52:58 2008	RSTPL_GetParentLevelObjects	Page 38 of 92
528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554	<pre>/* * Get Next Level Objects: * * This function is intended to allow retrieval of the children * of a given parent object. * * The caller specifies the parent object and * whether or not to include bad files. Even though the objects are * returned in a linked list, there could conceivably be thousands of * child objects, so the caller must specify the maximum number * of children to return. The caller is returned a cookie to allow * continuing on the next call to this function. * * Parameters: * context (I) - Pointer to the restore context * parentObj (I) - the parent object * objLevel (I) - specifies whether parentObject is a top level or * container object * objects (I) - a pointer to receive the start of the objects list * cookie (IO) - a place holder for the list position * maxEntries (I) - the maximum number of objects to return * numEntries (I) - the real number of objects returned in the list * allowBadFiles (I) - flag whether or not to include bad files * * Returns: * E_SUCCESS on success * EP_RB_RECOVER_XXX on error */ *****</pre>		547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000	<pre>errno_t RSTPL_GetParentLevelObjects(restore_context_t context, struct RSTPL_uro_list **parentObj, enum RSTPL_ObjectLevel objLevel, struct RSTPL_uro_list **objects, long *cookie, const long maxEntries, long *numEntries, const boolean_t allowBadFiles) { struct restoreContextData *rcd= (struct restoreContextData *) (context->appData); if (!rcd) return EP_RB_RECOVER_RC_APP_DATA_NULL; return RSTPL_GetParentLevelObjects(rcd, parentObj, objLevel, objects, cookie, maxEntries, numEntries, allowBadFiles); } *****</pre>	
Thu Jan 03 12:52:58 2008	restore_dcpdumpgcc 21	Page 37 of 92	Thu Jan 03 12:52:58 2008	restore_dcpdumpgcc 22	Page 38 of 92

Page 39 of 92	RSFPL_GainNextLevelObjects	Thu Jan 03 12:52:58 2008
647 1	//	
648 2	for(child:child->noder->getNextChild())	
649 3	//	
650 4	// skip this entry if we have chosen not to allow bad files	
651 5	//	
652 6	if (false == allowBadFiles) &&	
653 7	{	
654 8	if (false == allowBadFiles) &&	
655 9	continue;	
656 10	}	
657 11	continue;	
658 12	entries_count++;	
659 13	listentry=(RSRPC_uwo_list *)malloc(sizeof(
660 14	listentry->uwo_list *);	
661 15	struct NSRPC_uwo_list);	
662 16	if(!listentry)	
663 17	return EP_RB_RECOVER_MALLOC_FAILURE;	
664 18	return EP_RB_RECOVER_MALLOC_FAILURE;	
665 19	return EP_RB_RECOVER_MALLOC_FAILURE;	
666 20	return EP_RB_RECOVER_MALLOC_FAILURE;	
667 21	return EP_RB_RECOVER_MALLOC_FAILURE;	
668 22	return EP_RB_RECOVER_MALLOC_FAILURE;	
669 23	return EP_RB_RECOVER_MALLOC_FAILURE;	
670 24	return EP_RB_RECOVER_MALLOC_FAILURE;	
671 25	return EP_RB_RECOVER_MALLOC_FAILURE;	
672 26	return EP_RB_RECOVER_MALLOC_FAILURE;	
673 27	return EP_RB_RECOVER_MALLOC_FAILURE;	
674 28	return EP_RB_RECOVER_MALLOC_FAILURE;	
675 29	return EP_RB_RECOVER_MALLOC_FAILURE;	
676 30	return EP_RB_RECOVER_MALLOC_FAILURE;	
677 31	return EP_RB_RECOVER_MALLOC_FAILURE;	
678 32	return EP_RB_RECOVER_MALLOC_FAILURE;	
679 33	return EP_RB_RECOVER_MALLOC_FAILURE;	
680 34	return EP_RB_RECOVER_MALLOC_FAILURE;	
681 35	return EP_RB_RECOVER_MALLOC_FAILURE;	
682 36	return EP_RB_RECOVER_MALLOC_FAILURE;	
683 37	return EP_RB_RECOVER_MALLOC_FAILURE;	
684 38	return EP_RB_RECOVER_MALLOC_FAILURE;	
685 39	return EP_RB_RECOVER_MALLOC_FAILURE;	
686 40	return EP_RB_RECOVER_MALLOC_FAILURE;	
687 41	return EP_RB_RECOVER_MALLOC_FAILURE;	
688 42	return EP_RB_RECOVER_MALLOC_FAILURE;	
689 43	return EP_RB_RECOVER_MALLOC_FAILURE;	
690 44	return EP_RB_RECOVER_MALLOC_FAILURE;	
691 45	return EP_RB_RECOVER_MALLOC_FAILURE;	
692 46	return EP_RB_RECOVER_MALLOC_FAILURE;	
693 47	return EP_RB_RECOVER_MALLOC_FAILURE;	
694 48	return EP_RB_RECOVER_MALLOC_FAILURE;	
695 49	return EP_RB_RECOVER_MALLOC_FAILURE;	
696 50	return EP_RB_RECOVER_MALLOC_FAILURE;	
697 51	return EP_RB_RECOVER_MALLOC_FAILURE;	
698 52	return EP_RB_RECOVER_MALLOC_FAILURE;	
699 53	return EP_RB_RECOVER_MALLOC_FAILURE;	
700 54	return EP_RB_RECOVER_MALLOC_FAILURE;	
701 55	return EP_RB_RECOVER_MALLOC_FAILURE;	
702 56	return EP_RB_RECOVER_MALLOC_FAILURE;	
703 57	return EP_RB_RECOVER_MALLOC_FAILURE;	
704 58	return EP_RB_RECOVER_MALLOC_FAILURE;	
705 59	return EP_RB_RECOVER_MALLOC_FAILURE;	
706 60	return EP_RB_RECOVER_MALLOC_FAILURE;	
707 61	return EP_RB_RECOVER_MALLOC_FAILURE;	
708 62	return EP_RB_RECOVER_MALLOC_FAILURE;	
709 63	return EP_RB_RECOVER_MALLOC_FAILURE;	
710 64	return EP_RB_RECOVER_MALLOC_FAILURE;	
711 65	return EP_RB_RECOVER_MALLOC_FAILURE;	
712 66	return EP_RB_RECOVER_MALLOC_FAILURE;	
713 67	return EP_RB_RECOVER_MALLOC_FAILURE;	
714 68	return EP_RB_RECOVER_MALLOC_FAILURE;	
715 69	return EP_RB_RECOVER_MALLOC_FAILURE;	
716 70	return EP_RB_RECOVER_MALLOC_FAILURE;	
717 71	return EP_RB_RECOVER_MALLOC_FAILURE;	
718 72	return EP_RB_RECOVER_MALLOC_FAILURE;	
719 73	return EP_RB_RECOVER_MALLOC_FAILURE;	
720 74	return EP_RB_RECOVER_MALLOC_FAILURE;	
721 75	return EP_RB_RECOVER_MALLOC_FAILURE;	
722 76	return EP_RB_RECOVER_MALLOC_FAILURE;	
723 77	return EP_RB_RECOVER_MALLOC_FAILURE;	
724 78	return EP_RB_RECOVER_MALLOC_FAILURE;	
725 79	return EP_RB_RECOVER_MALLOC_FAILURE;	
726 80	return EP_RB_RECOVER_MALLOC_FAILURE;	
727 81	return EP_RB_RECOVER_MALLOC_FAILURE;	
728 82	return EP_RB_RECOVER_MALLOC_FAILURE;	
729 83	return EP_RB_RECOVER_MALLOC_FAILURE;	
730 84	return EP_RB_RECOVER_MALLOC_FAILURE;	
731 85	return EP_RB_RECOVER_MALLOC_FAILURE;	
732 86	return EP_RB_RECOVER_MALLOC_FAILURE;	
733 87	return EP_RB_RECOVER_MALLOC_FAILURE;	
734 88	return EP_RB_RECOVER_MALLOC_FAILURE;	
735 89	return EP_RB_RECOVER_MALLOC_FAILURE;	
736 90	return EP_RB_RECOVER_MALLOC_FAILURE;	
737 91	return EP_RB_RECOVER_MALLOC_FAILURE;	
738 92	return EP_RB_RECOVER_MALLOC_FAILURE;	
739 93	return EP_RB_RECOVER_MALLOC_FAILURE;	
740 94	return EP_RB_RECOVER_MALLOC_FAILURE;	
741 95	return EP_RB_RECOVER_MALLOC_FAILURE;	
742 96	return EP_RB_RECOVER_MALLOC_FAILURE;	
743 97	return EP_RB_RECOVER_MALLOC_FAILURE;	
744 98	return EP_RB_RECOVER_MALLOC_FAILURE;	
745 99	return EP_RB_RECOVER_MALLOC_FAILURE;	
746 100	return EP_RB_RECOVER_MALLOC_FAILURE;	
747 101	return EP_RB_RECOVER_MALLOC_FAILURE;	
748 102	return EP_RB_RECOVER_MALLOC_FAILURE;	
749 103	return EP_RB_RECOVER_MALLOC_FAILURE;	
750 104	return EP_RB_RECOVER_MALLOC_FAILURE;	
751 105	return EP_RB_RECOVER_MALLOC_FAILURE;	
752 106	return EP_RB_RECOVER_MALLOC_FAILURE;	
753 107	return EP_RB_RECOVER_MALLOC_FAILURE;	
754 108	return EP_RB_RECOVER_MALLOC_FAILURE;	
755 109	return EP_RB_RECOVER_MALLOC_FAILURE;	
756 110	return EP_RB_RECOVER_MALLOC_FAILURE;	
757 111	return EP_RB_RECOVER_MALLOC_FAILURE;	
758 112	return EP_RB_RECOVER_MALLOC_FAILURE;	
759 113	return EP_RB_RECOVER_MALLOC_FAILURE;	
760 114	return EP_RB_RECOVER_MALLOC_FAILURE;	
761 115	return EP_RB_RECOVER_MALLOC_FAILURE;	
762 116	return EP_RB_RECOVER_MALLOC_FAILURE;	
763 117	return EP_RB_RECOVER_MALLOC_FAILURE;	
764 118	return EP_RB_RECOVER_MALLOC_FAILURE;	
765 119	return EP_RB_RECOVER_MALLOC_FAILURE;	
766 120	return EP_RB_RECOVER_MALLOC_FAILURE;	
767 121	return EP_RB_RECOVER_MALLOC_FAILURE;	
768 122	return EP_RB_RECOVER_MALLOC_FAILURE;	
769 123	return EP_RB_RECOVER_MALLOC_FAILURE;	
770 124	return EP_RB_RECOVER_MALLOC_FAILURE;	
771 125	return EP_RB_RECOVER_MALLOC_FAILURE;	
772 126	return EP_RB_RECOVER_MALLOC_FAILURE;	

Page 40 of 92	RSFPL_GetNextLevelObjects	Thu Jan 03 12:52:58 2008
708 2	if (name)	
709 3	{	
710 4	rp(char *) (RnPropertyChar *) (name)->getValue();	
711 5	}	
712 6	}	
713 7	}	
714 8	}	
715 9	}	
716 10	}	
717 11	}	
718 12	}	
719 13	}	
720 14	}	
721 15	}	
722 16	}	
723 17	}	
724 18	}	
725 19	}	
726 20	}	
727 21	}	
728 22	}	
729 23	}	
730 24	}	
731 25	}	
732 26	}	
733 27	}	
734 28	}	
735 29	}	
736 30	}	
737 31	}	
738 32	}	
739 33	}	
740 34	}	
741 35	}	
742 36	}	
743 37	}	
744 38	}	
745 39	}	
746 40	}	
747 41	}	
748 42	}	
749 43	}	
750 44	}	
751 45	}	
752 46	}	
753 47	}	
754 48	}	
755 49	}	
756 50	}	
757 51	}	
758 52	}	
759 53	}	
760 54	}	
761 55	}	
762 56	}	
763 57	}	
764 58	}	
765 59	}	
766 60	}	
767 61	}	
768 62	}	
769 63	}	
770 64	}	
771 65	}	
772 66	}	
773 67	}	
774 68	}	
775 69	}	
776 70	}	
777 71	}	
778 72	}	
779 73	}	
780 74	}	
781 75	}	
782 76	}	
783 77	}	
784 78	}	
785 79	}	
786 80	}	
787 81	}	
788 82	}	
789 83	}	
790 84	}	
791 85	}	
792 86	}	
793 87	}	
794 88	}	
795 89	}	
796 90	}	
797 91	}	
798 92	}	
799 93	}	
800 94	}	
801 95	}	
802 96	}	
803 97	}	
804 98	}	
805 99	}	
806 100	}	
807 101	}	
808 102	}	
809 103	}	
810 104	}	
811 105	}	
812 106	}	
813 107	}	
814 108	}	
815 109	}	
816 110	}	
817 111	}	
818 112	}	
819 113	}	
820 114	}	
821 115	}	
822 116	}	
823 117	}	
824 118	}	
825 119	}	
826 120	}	
827 121	}	
828 122	}	
829 123	}	
830 124	}	
831 125	}	
832 126	}	
833 127	}	
834 128	}	
835 129	}	
836 130	}	
837 131	}	
838 132	}	
839 133	}	
840 134	}	
841 135	}	
842 136	}	
843 137	}	
844 138	}	
845 139	}	
846 140	}	
847 141	}	
848 142	}	
849 143	}	
850 144	}	
851 145	}	
852 146	}	
853 147	}	
854 148	}	
855 149	}	
856 150	}	
857 151	}	
858 152	}	
859 153	}	
860 154	}	
861 155	}	
862 156	}	
863 157	}	
864 158	}	
865 159	}	
866 160	}	
867 161	}	
868 162	}	
869 163	}	
870 164	}	
871 165	}	
872 166	}	
873 167	}	
874 168	}	
875 169	}	
876 170	}	
877 171	}	
878 172	}	
879 173	}	
880 174	}	
881 175	}	
882 176	}	
883 177	}	
884 178	}	
885 179	}	
886 180	}	
887 181	}	
888 182	}	
889 183	}	
890 184	}	
891 185	}	
892 186	}	
893 187	}	
894 188	}	
895 189	}	
896 190	}	
897 191	}	
898 192	}	
899 193	}	
900 194	}	
901 195	}	
902 196	}	
903 197	}	
904 198	}	
905 199	}	
906 200	}	
907 201	}	
908 202	}	
909 203	}	
910 204	}	
911 205	}	
912 206	}	
913 207	}	
914 208	}	
915 209	}	
916 210	}	
917 211	}	
918 212	}	
919 213	}	
920 214	}	
921 215	}	
922 216	}	
923 217	}	
924 218	}	
925 219	}	
926 220	}	
927 221	}	
928 222	}	
929 223	}	
930 224	}	
931 225	}	
932 226	}	
933 227	}	
934 228	}	
935 229	}	
936 230	}	
937 231	}	
938 232	}	
939 233	}	
940 234	}	
941 235	}	
942 236	}	
943 237	}	
944 238	}	
945 239	}	
946 240	}	
947 241	}	
948 242	}	
949 243	}	
950 244	}	
951 245	}	
952 246	}	
953 247	}	
954 248	}	
955 249	}	
956 250	}	
957 251	}	
958 252	}	
959 253	}	
960 254	}	
961 255	}	
962 256	}	
963 257	}	
964 258	}	
965 259	}	
966 260	}	
967 261	}	
968 262	}	
969 263	}	
970 264	}	
971 265	}	
972 266	}	
973 267	}	
974 268	}	
975 269	}	
976 270	}	
977 271	}	
978 272	}	
979 273	}	
980 274	}	
981 275	}	
982 276	}	
983 277	}	
984 278	}	
985 279	}	
986 280	}	
987 281	}	
988 282	}	
989 283	}	
990 284	}	
991 285	}	
992 286	}	
993 287	}	
994 288	}	
995 289	}	
996 290	}	
997 291	}	
998 292	}	
999 293	}	
1000 294	}	
1001 295	}	
1002 296	}	
1003 297	}	
1004 298	}	
1005 299	}	
1006 300	}	
1007 301	}	
1008 302	}	
1009 303	}	
1010 304	}	
1011 305	}	
1012 306	}	
1013 307	}	
1014 308	}	
1015 309	}	
1016 310	}	
1017 311	}	
1018 312	}	
1019 313	}	
1020 314	}	
1021 315	}	
1022 316	}	
1023 317	}	
1024 318	}	
1025 319	}	
1026 320	}	
1027 321	}	
1028 322	}	
1029 323	}	
1030 324	}	
1031 325	}	
1032 326	}	
1033 327	}	
1034 328	}	
1035 329	}	
1036 330	}	
1037 331	}	
1038 332	}	
1039 333	}	
1040 334	}	
1041 335	}	
1042 336	}	
1043 337	}	
1044 338	}	
1045 339	}	
1046 340	}	
1047 341	}	
1048 342	}	
1049 343	}	
1050 344	}	
1051 345	}	
1052 346	}	
1053 347	}	
1054 348	}	
1055 349	}	
1056 350	}	
1057 351	}	
1058 352	}	
1059 353	}	
1060 354	}	
1061 355	}	
1062 356	}	
1063 357	}	
1064 358	}	
1065 359	}	
1066 360	}	
1067 361	}	
1068 362	}	
1069 363	}	
1070 364	}	
1071 365	}	
1072 366	}	
1073 367	}	
1074 368	}	
1075 369	}	
1076 370	}	
1077 371	}	
1078 372	}	
1079 373	}	
1080 374	}	
1081 375	}	
1082 376	}	
1083 377	}	
1084 378	}	
1085 379	}	
1086 380	}	
1087 381	}	
1088 382	}	
1089 383	}	
1090 384	}	
1091 385	}	
1092 386	}	
1093 387	}	
1094 388	}	
1095 389	}	
1096 390	}	
1097 391	}	
1098 392	}	
1099 393	}	
1100 394	}	
1101 395	}	
1102 396	}	
1103 397	}	
1104 398	}	
1105 399	}	
1106 400	}	
1107 401	}	
1108 402	}	
1109 403	}	
1110 404	}	
1111 405	}	
1112 406	}	
1113 407	}	
1114 408	}	
1115 409	}	
1116 410	}	
1117 411	}	
1118 412	}	
1119 413	}	
1120 414	}	
1121 415	}	
1122 416	}	
1123 417	}	
1124 418	}	
1125 419	}	
1126 420	}	
1127 421	}	
1128 422	}	
1129 423	}	
1130 424	}	
1131 425	}	
1132 426	}	
1133 427	}	
1134 428	}	
1135 429	}	
1136 430	}	
1137 431	}	
1138 432	}	
1139 433	}	
1140 434	}	
1141 435	}	
1142 436	}	
1143 437	}	
1144 438	}	
1145 439	}	
1146 440	}	
1147 441	}	
1148 442	}	
1149 443	}	
1150 444	}	
1151 445	}	
1152 446	}	
1153 447	}	
1154 448	}	
1155 449	}	
1156 450	}	
1157 451	}	
1158 452	}	
1159 453	}	
1160 454	}	
1161 455	}	

```

773 2         if (NULL==uobj->objOwnerName || NULL==uobj->objGroupname)
774 3             {
775 4                 rthe_log_stats(
                        0, "plugin.cc - allocation failure from esi_strdup");
                        return ERR_RB_RECOVER_MALLOC_FAILURE;
                    }
776 3             }
777 2
778 2             RbPropertyDate *pr = (RbPropertyDate *) (child->getProperty(
779 3                 PROPERTY_BACKUP_DATE));
780 2             if (pr)
781 2             {
782 3                 uobj->objModTime=pr->getValue()->unixTime();
783 3             }
784 2             else
785 2             {
786 3                 uobj->objModTime=0;
787 3                 RbProperty *bnode = (RbProperty *) (child->getBackupNodePtr());
788 3                 if (bnode)
789 4                 {
790 4                     pr = (RbPropertyDate *) (bnode->getProperty(
791 5                         PROPERTY_BACKUP_DATE));
792 4                     if (pr)
793 5                     {
794 5                         uobj->objModTime=pr->getValue()->unixTime();
795 5                     }
796 4                 }
797 2             }
798 2
799 2             uobj->objSize_high=child->getSize().high;
800 2             uobj->objSize_low=child->getSize().low;
801 2
802 2             if (child->getStatebit(STATE_BADFILE))
803 2             {
804 3                 uobj->objBackupStatus=RSTRPC_Backup_Bad;
805 2             }
806 2             else if ( child->getStatebit(STATE_INVALID_SSID |
807 2                 STATE_NO_SSID |
808 2                 STATE_EXPIRED_SSID ) )
809 3             {
810 3                 uobj->objBackupStatus=RSTRPC_Backup_Expired;
811 2             }
812 2             else if ( child->getStatebit(STATE_CH_INVALID_SSID |
813 2                 STATE_CH_NO_SSID |
814 2                 STATE_CH_EXPIRED_SSID |
815 2                 STATE_CH_BADFILE ) )
816 3             {
817 3                 uobj->objBackupStatus=RSTRPC_Backup_Child_Without_Data;
818 2             }
819 2             else
820 2             {
821 3                 uobj->objBackupStatus=RSTRPC_Backup_Good;
822 2             }
823 2
824 2             listentry->next=NULL;
825 2             if (previous)
826 2             {
827 3                 previous->next=listentry;
828 2             }
829 2             else
830 2             {
831 3                 objects=listentry;
832 2             }
833 2             previous=listentry;
834 2

```

```

835 2             if (entries_count==maxEntries)
836 3             {
837 4                 *cookie=(log)child;
838 3                 *numberEntries=entries_count;
839 3                 return E_SUCCESS;
840 3             }
841 2             }
842 2             }
843 2
844 1             *cookie=DONE_COOKIE;
845 1             *numberEntries=entries_count;
846 1             return E_SUCCESS;
847 1

```

```

831 //
832 * Mark Object
833 *
834 * The MarkObject operation takes a restorableObject and marks it,
835 * possibly its descendant files for restoration based on the input
836 * criteria.
837 * Since the RSTPL_MarkObject call is an asynchronously executed
838 * operation, in the Restore Engine that performs the marking,
839 * periodically check for user-signalled cancellation,
840 * and update progress
841 * data using the progress callback function argument.
842 *
843 * NOTE: This function is responsible for keeping the volumes needed
844 * (by/for) element of the restore context up to date.
845 *
846 * Parameters:
847 * context (I) - Pointer to the restore context
848 * thisObject (I) - The restorable object;
849 * file, or a container object (e.g., a directory).
850 *
851 * allowBadFiles (I) - allows marking of files of state BADDATA.
852 * descend (I) - Should mark operation descend to operate on the content
853 * of container objects.
854 * BadFilesCount (O) - returns the file count with BADDATA.
855 * PermBadFilesCount (O) - returns the file count with permission denied.
856 * fileMarked (O) - returns the total files marked after this mark occurred.
857 * lenMarkedFiles (O) - return the length of files marked after this mark
858 * occurred.
859 * otherMarked (O) - return the total directories marked after this mark
860 * occurred.
861 * progressCB (I) - pointer to callback function to report progress and
862 * test for cancellation
863 *
864 * Note:
865 * The Progress Callback is currently not used. If this need
866 * for this development arises, then use this argument
867 * (mark, unmark, submark) must be enhanced.
868 *
869 *
870 */
871
872 ...../
873
874 ...../
875
876 ...../
877
878 ...../
879
880 ...../
881
882 ...../
883
884 ...../
885
886 ...../
887
888 ...../
889
890 ...../
891
892 ...../
893
894 ...../
895
896 ...../
897
898 ...../
899
900 ...../
901
902 ...../
903
904 ...../
905
906 ...../
907
908 ...../
909
910 ...../
911
912 ...../
913
914 ...../
915
916 ...../
917
918 ...../
919
920 ...../
921
922 ...../
923
924 ...../
925
926 ...../
927
928 ...../
929
930 ...../
931
932 ...../
933
934 ...../
935
936 ...../
937
938 ...../
939
940 ...../
941
942 ...../
943
944 ...../
945
946 ...../
947
948 ...../
949
950 ...../
951
952 ...../
953
954 ...../
955
956 ...../
957
958 ...../
959
960 ...../
961
962 ...../
963
964 ...../
965
966 ...../
967
968 ...../
969
970 ...../
971
972 ...../
973
974 ...../
975
976 ...../
977
978 ...../
979
980 ...../
981
982 ...../
983
984 ...../
985
986 ...../
987
988 ...../
989
990 ...../
991
992 ...../
993
994 ...../
995
996 ...../
997
998 ...../
999
1000 ...../

```

```

691 restore_by_RSTPL_MarkObject(restore_context *context,
692 struct RSTPL_User_Restorable_Object *thisObject,
693 boolean_t allowBadFiles,

```

```

995 boolean_t descend,
996 unsigned long *badFilesCount,
997 unsigned long *permBadFilesCount,
998 unsigned long *fileMarked,
999 unsigned long *lenMarkedFiles,
1000 u_int32_t *mark, u_int32_t *submark);

```

```

900      unsigned long *dlistMarked
901      unsigned long *otherMarked
902      RSTPI_MarkProgressProc progressCB)
903  {
904      char ** c1=(char **) (thisObject->appdata.data);
905      RestoreNode *rnodep=(RestoreNode *) (*c1);
906      if (NULL==rnodep)
907      {
908          the_log_stats(
909              0,"RSTPI_MarkObject - Mark object has no app data");
910      }
911      return EP_RB_RECOVER_RV_APP_DATA_NULL;
912  }
913
914  // MTFB (multi trail) file backup nodes are to be treated as atomic.
915  // Each MTFB node is trying to mark its a child of MTFB node. When
916  // we perform the mark operation on the entire MTFB node.
917  //
918  RestoreNode *parent=rnodep->getparent();
919  if (NULL !=parent && parent->nodeType() == RNC_MTFB)
920  {
921      rnodep=parent;
922  }
923
924  *badFilesCount=0;
925  *pendingFilesCount=0;
926  *filesMarked=0;
927  *lenMarkedFiles=ul_to_ah(0);
928  *dirMarked=0;
929  *otherMarked=0;
930
931  rnodep->markNode(descend,allowBadFiles);
932
933  // We start at the backup node pointer since we want the total
934  // for the entire backup, not just this node down (except the
935  // badFilesCount, in which case the affected nodes will only
936  // be from this level down)
937
938  *filesMarked=rnodep->getBackupNodePtr()->countMarkedNodes(
939      *lenMarkedFiles=rnodep->getBackupNodePtr()->getCalMarkedSize(),
940      RNC_ANY_DATAFILE);
941
942  // The bad file count is a count of the bad files encountered
943  // (not necessarily marked) from the current node down.
944
945  *badFilesCount=rnodep->countMarkedBadFilesNodes(RNC_ANY_DATAFILE);
946
947  return E_SUCCESS;
948  }

```

930 //

Page 47 of 92	RSTPL_UnmarkObject	Thu Jan 03 12:52:58 2008
931	/*	1061 1
932	* UnmarkObject	1062 1
933	* The UnmarkObject operation takes a restoreableObject and unmarks	1064 1
934	* possibly its descendant files for restoreal based on the input	1065 1
935	* Since the RSTPL_UnmarkObject call is an asynchronously executed	1067 1
936	* operation	1068 2
937	* in the restore engine that performs the unmarking,	1069 2
938	* must periodically check for user-signalized cancellation and update	1010 1
939	* progress data using the progress callback function argument.	1011 1
940	* NOTE: This function is responsible for keeping the volumes needed	1013 1
941	* list	1014 1
942	* (byValue) element of the restore context up to date.	1015 2
943	* UnmarkObject Parameters:	1016 2
944	* context (I) - pointer to the restore context	1017 1
945	* thisObject (I) - The restoreal object;	1018 1
946	* (can be a leaf object (e.g. a	1019 1
947	* file), or a container object (1020 1
948	* e.g., a directory).	1021 1
949	* BadFileOnly (I) - allows unmarking ONLY of files of state BADDATA.	1022 1
950	* descend (I) - Should unmark operation descend to operate on the	1025 1
951	* content of container objects.	1027 1
952	* BadFileCount (O) - returns the file count with BADDATA.	1028 1
953	* fileMarked (O) - return the total files marked after this unmark	1029 1
954	* leMarkedFile (O) - return the total files marked after this unmark	1030 1
955	* fileMarked (O) - return the length of files marked after this unmark	1031 1
956	* fileMarked (O) - return the total directories marked after this unmark	1032 1
957	* otherMarked (O) - return the total "other" files marked after this unmark	1033 1
958	* progress (I) - pointer to callback function to report progress and	1034 1
959	* test for cancellation	1035 1
960	* Note: The Progress Callback is currently not used. If the need	1036 1
961	* for this developer, the routines which use this argument	1037 1
962	* (mark, unmark, submit) must be enhanced.	1038 1
963	*	1039 1
964	*	1040 1
965	*	1041 1
966	*	1042 1
967	*	1043 1
968	*	1044 1
969	*	1045 1
970	*	1046 1
971	*	1047 1
972	*	1048 1
973	*	1049 1
974	*	1050 1
975	*	1051 1
976	*	1052 1
977	*	1053 1
978	*	1054 1
979	*	1055 1
980	*	1056 1
981	*	1057 1
982	*	1058 1
983	*	1059 1
984	*	1060 1
985	*	1061 1
986	*	1062 1
987	*	1063 1
988	*	1064 1
989	*	1065 1
990	*	1066 1
991	*	1067 1
992	*	1068 1
993	*	1069 1
994	*	1070 1
995	*	1071 1
996	*	1072 1
997	*	1073 1
998	*	1074 1
999	*	1075 1
1000	*	1076 1

```

1046 //
1047
1048 * * * Is Object Markable
1049 * * This function determines if a restorable object has been
1050 * * marked for restoration.
1051 * * It is intended to allow the user to determine the
1052 * * current restore markings for the restorable objects at the same
1053 * * hierarchy level.
1054 * * i.e. objects that have the same parent restorableobject.
1055 * * Parameters:
1056 * * context { I1 } - Pointer to the restore context
1057 * * thisObject { I1 } - The restoral object to be checked; can be a leaf object
1058 * * (e.g. a file), or a container (e.g. a directory).
1059 * * markable { O } - boolean to receive the marked(I1) / umarked(I1) result
1060
1061 ...../
1062
1063 .....by RSTPL_IsObjectMarkable(restore_context *context,
1064 struct RSTPL_User_Restorable) a object
1065 .....by markable
1066 {
1067     RestoreNode *rnodep = *(RestoreNode **) (thisObject->appData.data);
1068
1069     if (NULL==rnodep)
1070     {
1071         return EP_RB_RECOVER_IN_APP_DATA_NULL;
1072     }
1073     *markable=rnodep->isNodeMarkable();
1074     return E_SUCCESS;
1075 }
1076
1077
1078
1079

```

```

1082 //
1083 //*****
1084 * Is Object Marked
1085 * This function determines if a restorable object has been
1086 * marked for restoration.
1087 * It is intended to allow the user to determine the
1088 * current restore markings for the restorable objects at the same
1089 * object tree
1090 * hierarchy level
1091 * i.e. objects that have the same parent restorableobject.
1092 * Parameters:
1093 * context (I) - Pointer to the restore context
1094 * thisObject (I) - The restoral object to be checked; can be a leaf object
1095 * (e.g. a file), or a container object (e.g. a directory).
1096 * marked (O) - boolean to receive the marked(I) / unmarked(I) result
1097 *****/
1098
1099 errno_t RSTPL_IsObjectMarked(RestoreContext *context,
1100 struct RSTPL_User_RestorableObject *thisObject,
1101 boolean_Ly *marked)
1102 {
1103     RestoreNode *rnodep = *(RestoreNode **) (thisObject->appdata.data);
1104
1105     if (NULL == rnodep)
1106     {
1107         return EP_RL_RECOVER_RN_APP_DATA_NULL;
1108     }
1109
1110     *marked = rnodep->isNodeMarked();
1111
1112     return R_SUCCESS;
1113 }
1114
1115

```

1117 //

```

1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3
```



```

1180 2         }
1181 2         // Get Unix flavored time
1183 2         DateTime *dt=(DateTime *) (p->getValue());
1185 2         time_t backupTime=dt->unixTime();
1187 2         // time_t backupTime=(DateTime) (p->getValue())->unixTime();
1188 2         // Get all dates if start and end time are zero
1189 2         // otherwise make sure the time is in the selected range
1190 2         // Also, get complete backups ONLY if that is what is required
1191 2         //
1192 2         if (
1193 2             ( (secTime==0) && (endTime==0) ) ||
1194 2             (startTime<backupTime) && (endTime>backupTime)
1195 2             )
1196 2             &&
1197 2             ( (flag==BACKUP_SELECTION_FLAG_PARTIAL_OK) ||
1198 2               bn->getStateBit(STATE_COMPLETE)
1199 2             )
1200 2         )
1201 2         {
1202 2             //
1203 2             // Ok, we have found a backup time within the
1204 2             // range. Add it here.
1205 2             RSTRPC_time_list *newTime=(RSTRPC_time_list *) malloc(
1206 2                 sizeof(struct RSTRPC_time_list));
1207 2             if(newTime)
1208 2             {
1209 2                 the_log_state(0,"plugin.cc - malloc failure");
1210 2                 return E_PLUGIN_RECOVER_FAILURE;
1211 2             }
1212 2             //
1213 2             // Increment count of valid time entries
1214 2             {numEntries}++;
1215 2             newTime->next=NULL;
1216 2             newTime->time=backupTime;
1217 2             if(newList)
1218 2             {
1219 2                 newList->next=newTime;
1220 2                 newList=newTime;
1221 2             }
1222 2             if(flag)
1223 2             {
1224 2                 *timeList=newList;
1225 2                 first=0;
1226 2             }
1227 2         }
1228 2     }
1229 2     return E_SUCCESS;
1230 2 }
1231 2
1232 2
1233 2
1234 2
1235 2
1236 2
1237 2
1238 2
1239 2

```

```

1241 //

```

Thu Jan 03 12:52:58 2008	RSTPL_GetCurrentBackupTime	Page 57 of 92	Thu Jan 03 12:52:58 2008	Page 58 of 92
1242	1247	
1243	1248	
1244	1249	
1245	1250	
1246	1251	
1247	1252	
1248	1253	
1249	1254	
1250	1255	
1251	1256	
1252	1257	
1253	1258	
1254	1259	
1255	1260	
1256	1261	
1257	1262	
1258	1263	
1259	1264	
1260	1265	
1261	1266	
1262	1267	
1263	1268	
1264	1269	
1265	1270	
1266	1271	
1267	1272	
1268	1273	
1269	1274	
1270	1275	
1271	1276	
1272	1277	
1273	1278	
1274	1279	
1275	1280	
1276	1281	
1277	1282	
1278	1283	
1279	1284	
1280	1285	
1281	1286	
1282	1287	
1283	1288	
1284	1289	
1285	1290	
1286	1291	
1287	1292	
1288	1293	
1289	1294	
1290	1295	
1291	1296	
1292	1297	
1293	1298	
1294	1299	
1295	1300	
1296	1301	
1297	1302	
1298	1303	
1299	1304	
1300	1305	
1301	1306	
1302	1307	
1303	1308	
1304	1309	
1305	1310	
1306	1311	
1307	1312	
1308	1313	
1309	1314	
1310	1315	
1311	1316	
1312	1317	
1313	1318	
1314	1319	
1315	1320	
1316	1321	
1317	1322	
1318	1323	
1319	1324	
1320	1325	
1321	1326	
1322	1327	
1323	1328	
1324	1329	
1325	1330	
1326	1331	
1327	1332	
1328	1333	
1329	1334	
1330	1335	
1331	1336	
1332	1337	
1333	1338	
1334	1339	
1335	1340	
1336	1341	
1337	1342	
1338	1343	
1339	1344	
1340	1345	
1341	1346	
1342	1347	
1343	1348	
1344	1349	
1345	1350	
1346	1351	
1347	1352	
1348	1353	
1349	1354	
1350	1355	
1351	1356	
1352	1357	
1353	1358	
1354	1359	
1355	1360	
1356	1361	
1357	1362	
1358	1363	
1359	1364	
1360	1365	
1361	1366	
1362	1367	
1363	1368	
1364	1369	
1365	1370	
1366	1371	
1367	1372	
1368	1373	
1369	1374	
1370	1375	
1371	1376	
1372	1377	
1373	1378	
1374	1379	
1375	1380	
1376	1381	
1377	1382	
1378	1383	
1379	1384	
1380	1385	
1381	1386	
1382	1387	
1383	1388	
1384	1389	
1385	1390	
1386	1391	
1387	1392	
1388	1393	
1389	1394	
1390	1395	
1391	1396	
1392	1397	
1393	1398	
1394	1399	
1395	1400	
1396	1401	
1397	1402	
1398	1403	
1399	1404	
1400	1405	
1401	1406	
1402	1407	
1403	1408	
1404	1409	
1405	1410	
1406	1411	
1407	1412	
1408	1413	
1409	1414	
1410	1415	
1411	1416	
1412	1417	
1413	1418	
1414	1419	
1415	1420	
1416	1421	
1417	1422	
1418	1423	
1419	1424	
1420	1425	
1421	1426	
1422	1427	
1423	1428	
1424	1429	
1425	1430	
1426	1431	
1427	1432	
1428	1433	
1429	1434	
1430	1435	
1431	1436	
1432	1437	
1433	1438	
1434	1439	
1435	1440	
1436	1441	
1437	1442	
1438	1443	
1439	1444	
1440	1445	
1441	1446	
1442	1447	
1443	1448	
1444	1449	
1445	1450	
1446	1451	
1447	1452	
1448	1453	
1449	1454	
1450	1455	
1451	1456	
1452	1457	
1453	1458	
1454	1459	
1455	1460	
1456	1461	
1457	1462	
1458	1463	
1459	1464	
1460	1465	
1461	1466	
1462	1467	
1463	1468	
1464	1469	
1465	1470	
1466	1471	
1467	1472	
1468	1473	
1469	1474	
1470	1475	
1471	1476	
1472	1477	
1473	1478	
1474	1479	
1475	1480	
1476	1481	
1477	1482	
1478	1483	
1479	1484	
1480	1485	
1481	1486	
1482	1487	
1483	1488	
1484	1489	
1485	1490	
1486	1491	
1487	1492	
1488	1493	
1489	1494	
1490	1495	
1491	1496	
1492	1497	
1493	1498	
1494	1499	
1495	1500	
1496	1501	
1497	1502	
1498	1503	
1499	1504	
1500	1505	
1501	1506	
1502	1507	
1503	1508	
1504	1509	
1505	1510	
1506	1511	
1507	1512	
1508	1513	
1509	1514	
1510	1515	
1511	1516	
1512	1517	
1513	1518	
1514	1519	
1515	1520	
1516	1521	
1517	1522	
1518	1523	
1519	1524	
1520	1525	
1521	1526	
1522	1527	
1523	1528	
1524	1529	
1525	1530	
1526	1531	
1527	1532	
1528	1533	
1529	1534	
1530	1535	
1531	1536	
1532	1537	
1533	1538	
1534	1539	
1535	1540	
1536	1541	
1537	1542	
1538	1543	
1539	1544	
1540	1545	
1541	1546	
1542	1547	
1543	1548	
1544	1549	
1545	1550	
1546	1551	
1547	1552	
1548	1553</		

Page 59 of 92	RSPL_SelBackupForTime	Thu Jan 03 12:52:58 2008	Page 60 of 92	RSPL_SelBackupForTime	Thu Jan 03 12:52:58 2008
<pre>1298 /***** 1299 * 1300 * Set Backup For Time 1301 * 1302 * Function Description: 1303 * Switch to the backup plane of the specified time, 1304 * that is before the specified time, 1305 * or the most recent 1306 * If an exact match is not possible. 1307 * 1308 * Parameters: 1309 * context (t) - Pointer to the restore context 1310 * ForTime (t) - The time for which the backup is requested 1311 * flags (f) - Backup constraint flags: e.g., full-only/partial-ok 1312 * 1313 * Return Codes: 1314 * E_SUCCESS - operation completed successfully 1315 * E_NOTRECOVERABLE - backup plane cannot be found 1316 * 1317 *****/ 1318 extern t RSPL_SelBackupForTime(t restore_context *context, 1319 const time_t forTime, 1320 RSPLPC_Backup_Flags_t flags) 1321 { 1322 struct restoreContextData *rcd=(struct restoreContextData *) (1323 context->appData); 1324 1325 if(!rcd) 1326 { 1327 return E_RB_RECOVER_RC_APP_DATA_NULL; 1328 } 1329 1330 MSetNode *w_list=rcd->currentWsetNode; 1331 if(!w_list) 1332 { 1333 return E_RB_RECOVER_RM_APP_DATA_NULL; 1334 } 1335 1336 // Loop through each one on the current list and return the one 1337 // are looking for. 1338 for(RestoreNode *bn=w_list->getNextChild(1); bn; bn=w_list->getNextChild(1)) 1339 { 1340 RbPropertyData *pr=(RbPropertyData *) (bn->getPropertiy(1341 PROPERTY_BACKUP_DATE)); 1342 if(!pr) 1343 { 1344 continue; 1345 } 1346 // Get Unix flavored time 1347 Datetime *dt=(Datetime *) (pr->getValue(1)); 1348 time_t backupTime=dt->unixTime(); 1349 if ((forTime >= backupTime) && 1350 (1351</pre>	<pre>1357 2 1358 3 1359 3 1360 3 1361 3 1362 3 1363 3 1364 3 1365 4 1366 4 1367 4 1368 4 1369 4 1370 4 1371 4 1372 4 1373 3 1374 3 1375 3 1376 4 1377 4 1378 4 1379 3 1380 3 1381 3 1382 2 1383 1 1384 1 1385 1 1386 1 1387 1 1388 1 1389 1 1390 1 1391 1 1392 1 1393 1 1394 1 1395 1 1396 1 1397 1 1398 1 1399 1 1400 1 1401 1 1402 1 1403 1 1404 1 1405 1 1406 1 1407 1 1408 1 1409 1 1410 1 1411 1 1412 1 1413 1 1414 1 1415 1 1416 1 1417 1 1418 1 1419 1 1420 1 1421 1 1422 1 1423 1 1424 1 1425 1 1426 1 1427 1 1428 1 1429 1 1430 1 1431 1 1432 1 1433 1 1434 1 1435 1 1436 1 1437 1 1438 1 1439 1 1440 1 1441 1 1442 1 1443 1 1444 1 1445 1 1446 1 1447 1 1448 1 1449 1 1450 1 1451 1 1452 1 1453 1 1454 1 1455 1 1456 1 1457 1 1458 1 1459 1 1460 1 1461 1 1462 1 1463 1 1464 1 1465 1 1466 1 1467 1 1468 1 1469 1 1470 1 1471 1 1472 1 1473 1 1474 1 1475 1 1476 1 1477 1 1478 1 1479 1 1480 1 1481 1 1482 1 1483 1 1484 1 1485 1 1486 1 1487 1 1488 1 1489 1 1490 1 1491 1 1492 1 1493 1 1494 1 1495 1 1496 1 1497 1 1498 1 1499 1 1500 1 1501 1 1502 1 1503 1 1504 1 1505 1 1506 1 1507 1 1508 1 1509 1 1510 1 1511 1 1512 1 1513 1 1514 1 1515 1 1516 1 1517 1 1518 1 1519 1 1520 1 1521 1 1522 1 1523 1 1524 1 1525 1 1526 1 1527 1 1528 1 1529 1 1530 1 1531 1 1532 1 1533 1 1534 1 1535 1 1536 1 1537 1 1538 1 1539 1 1540 1 1541 1 1542 1 1543 1 1544 1 1545 1 1546 1 1547 1 1548 1 1549 1 1550 1 1551 1 1552 1 1553 1 1554 1 1555 1 1556 1 1557 1 1558 1 1559 1 1560 1 1561 1 1562 1 1563 1 1564 1 1565 1 1566 1 1567 1 1568 1 1569 1 1570 1 1571 1 1572 1 1573 1 1574 1 1575 1 1576 1 1577 1 1578 1 1579 1 1580 1 1581 1 1582 1 1583 1 1584 1 1585 1 1586 1 1587 1 1588 1 1589 1 1590 1 1591 1 1592 1 1593 1 1594 1 1595 1 1596 1 1597 1 1598 1 1599 1 1600 1 1601 1 1602 1 1603 1 1604 1 1605 1 1606 1 1607 1 1608 1 1609 1 1610 1 1611 1 1612 1 1613 1 1614 1 1615 1 1616 1 1617 1 1618 1 1619 1 1620 1 1621 1 1622 1 1623 1 1624 1 1625 1 1626 1 1627 1 1628 1 1629 1 1630 1 1631 1 1632 1 1633 1 1634 1 1635 1 1636 1 1637 1 1638 1 1639 1 1640 1 1641 1 1642 1 1643 1 1644 1 1645 1 1646 1 1647 1 1648 1 1649 1 1650 1 1651 1 1652 1 1653 1 1654 1 1655 1 1656 1 1657 1 1658 1 1659 1 1660 1 1661 1 1662 1 1663 1 1664 1 1665 1 1666 1 1667 1 1668 1 1669 1 1670 1 1671 1 1672 1 1673 1 1674 1 1675 1 1676 1 1677 1 1678 1 1679 1 1680 1 1681 1 1682 1 1683 1 1684 1 1685 1 1686 1 1687 1 1688 1 1689 1 1690 1 1691 1 1692 1 1693 1 1694 1 1695 1 1696 1 1697 1 1698 1 1699 1 1700 1 1701 1 1702 1 1703 1 1704 1 1705 1 1706 1 1707 1 1708 1 1709 1 1710 1 1711 1 1712 1 1713 1 1714 1 1715 1 1716 1 1717 1 1718 1 1719 1 1720 1 1721 1 1722 1 1723 1 1724 1 1725 1 1726 1 1727 1 1728 1 1729 1 1730 1 1731 1 1732 1 1733 1 1734 1 1735 1 1736 1 1737 1 1738 1 1739 1 1740 1 1741 1 1742 1 1743 1 1744 1 1745 1 1746 1 1747 1 1748 1 1749 1 1750 1 1751 1 1752 1 1753 1 1754 1 1755 1 1756 1 1757 1 1758 1 1759 1 1760 1 1761 1 1762 1 1763 1 1764 1 1765 1 1766 1 1767 1 1768 1 1769 1 1770 1 1771 1 1772 1 1773 1 1774 1 1775 1 1776 1 1777 1 1778 1 1779 1 1780 1 1781 1 1782 1 1783 1 1784 1 1785 1 1786 1 1787 1 1788 1 1789 1 1790 1 1791 1 1792 1 1793 1 1794 1 1795 1 1796 1 1797 1 1798 1 1799 1 1800 1 1801 1 1802 1 1803 1 1804 1 1805 1 1806 1 1807 1 1808 1 1809 1 1810 1 1811 1 1812 1 1813 1 1814 1 1815 1 1816 1 1817 1 1818 1 1819 1 1820 1 1821 1 1822 1 1823 1 1824 1 1825 1 1826 1 1827 1 1828 1 1829 1 1830 1 1831 1 1832 1 1833 1 1834 1 1835 1 1836 1 1837 1 1838 1 1839 1 1840 1 1841 1 1842 1 1843 1 1844 1 1845 1 1846 1 1847 1 1848 1 1849 1 1850 1 1851 1 1852 1 1853 1 1854 1 1855 1 1856 1 1857 1 1858 1 1859 1 1860 1 1861 1 1862 1 1863 1 1864 1 1865 1 1866 1 1867 1 1868 1 1869 1 1870 1 1871 1 1872 1 1873 1 1874 1 1875 1 1876 1 1877 1 1878 1 1879 1 1880 1 1881 1 1882 1 1883 1 1884 1 1885 1 1886 1 1887 1 1888 1 1889 1 1890 1 1891 1 1892 1 1893 1 1894 1 1895 1 1896 1 1897 1 1898 1 1899 1 1900 1 1901 1 1902 1 1903 1 1904 1 1905 1 1906 1 1907 1 1908 1 1909 1 1910 1 1911 1 1912 1 1913 1 1914 1 1915 1 1916 1 1917 1 1918 1 1919 1 1920 1 1921 1 1922 1 1923 1 1924 1 1925 1 1926 1 1927 1 1928 1 1929 1 1930 1 1931 1 1932 1 1933 1 1934 1 1935 1 1936 1 1937 1 1938 1 1939 1 1940 1 1941 1 1942 1 1943 1 1944 1 1945 1 1946 1 1947 1 1948 1 1949 1 1950 1 1951 1 1952 1 1953 1 1954 1 1955 1 1956 1 1957 1 1958 1 1959 1 1960 1 1961 1 1962 1 1963 1 1964 1 1965 1 1966 1 1967 1 1968 1 1969 1 1970 1 1971 1 1972 1 1973 1 1974 1 1975 1 1976 1 1977 1 1978 1 1979 1 1980 1 1981 1 1982 1 1983 1 1984 1 1985 1 1986 1 1987 1 1988 1 1989 1 1990 1 1991 1 1992 1 1993 1 1994 1 1995 1 1996 1 1997 1 1998 1 1999 1 2000 1 2001 1 2002 1 2003 1 2004 1 2005 1 2006 1 2007 1 2008 1 2009 1 2010 1 2011 1 2012 1 2013 1 2014 1 2015 1 2016 1 2017 1 2018 1 2019 1 2020 1 2021 1 2022 1 2023 1 2024 1 2025 1 2026 1 2027 1 2028 1 2029 1 2030 1 2031 1 2032 1 2033 1 2034 1 2035 1 2036 1 2037 1 2038 1 2039 1 2040 1 2041 1 2042 1 2043 1 2044 1 2045 1 2046 1 2047 1 2048 1 2049 1 2050 1 2051 1 2052 1 2053 1 2054 1 2055 1 2056 1 2057 1 2058 1 2059 1 2060 1 2061 1 2062 1 2063 1 2064 1 2065 1 2066 1 2067 1 2068 1 2069 1 2070 1 2071 1 2072 1 2073 1 2074 1 2075 1 2076 1 2077 1 2078 1 2079 1 2080 1 2081 1 2082 1 2083 1 2084 1 2085 1 2086 1 2087 1 2088 1 2089 1 2090 1 2091 1 2092 1 2093 1 2094 1 2095 1 2096 1 2097 1 2098 1 2099 1 2100 1 2101 1 2102 1 2103 1 2104 1 2105 1 2106 1 2107 1 2108 1 2109 1 2110 1 2111 1 2112 1 2113 1 2114 1 2115 1 2116 1 2117 1 2118 1 2119 1 2120 1 2121 1 2122 1 2123 1 2124 1 2125 1 2126 1 2127 1 2128 1 2129 1 2130 1 2131 1 2132 1 2133 1 2134 1 2135 1 2136 1 2137 1 2138 1 2139 1 2140 1 2141 1 2142 1 2143 1 2144 1 2145 1 2146 1 2147 1 2148 1 2149 1 2150 1 2151 1 2152 1 2153 1 2154 1 2155 1 2156 1 2157 1 2158 1 2159 1 2160 1 2161 1 2162 1 2163 1 2164 1 2165 1 2166 1 2167 1 2168 1 2169 1 2170 1 2171 1 2172 1 2173 1 2174 1 2175 1 2176 1 2177 1 2178 1 2179 1 2180 1 2181 1 2182 1 2183 1 2184 1 2185 1 2186 1 2187 1 2188 1 2189 1 2190 1 2191 1 2192 1 2193 1 2194 1 2195 1 2196 1 2197 1 2198 1 2199 1 2200 1 2201 1 2202 1 2203 1 2204 1 2205 1 2206 1 2207 1 2208 1 2209 1 2210 1 2211 1 2212 1 2213 1 2214 1 2215 1 2216 1 2217 1 2218 1 2219 1 2220 1 2221 1 2222 1 2223 1 2224 1 2225 1 2226 1 2227 1 2228 1 2229 1 2230 1 2231 1 2232 1 2233 1 2234 1 2235 1 2236 1 2237 1 2238 1 2239 1 2240 1 2241 1 2242 1 2243 1 2244 1 2245 1 2246 1 2247 1 2248 1 2249 1 2250 1 2251 1 2252 1 2253 1 2254 1 2255 1 2256 1 2257 1 2258 1 2259 1 2260 1 2261 1 2262 1 2263 1 2264 1 2265 1 2266 1 2267 1 2268 1 2269 1 2270 1 2271 1 2272 1 2273 1 2274 1 2275 1 2276 1 2277 1 2278 1 2279 1 2280 1 2281 1 2282 1 2283 1 2284 1 2285 1 2286 1 2287 1 2288 1 2289 1 2290 1 2291 1 2292 1 2293 1 2294 1 2295 1 2296 1 2297 1 2298 1 2299 1 2300 1 2301 1 2302 1 2303 1 2304 1 2305 1 2306 1 2307 1 2308 1 2309 1 2310 1 2311 1 2312 1 2313 1 2314 1 2315 1 2316 1 2317 1 2318 1 2319 1 2320 1 2321 1 2322 1 2323 1 2324 1 2325 1 2326 1 2327 1 2328 1 2329 1 2330 1 2331 1 2332 1 2333 1 2334 1 2335 1 2336 1 2337 1 2338 1 2339 1 2340 1 2341 1 2342 1 2343 1 2344 1 2345 1 2346 1 2347 1 2348 1 2349 1 2350 1 2351 1 2352 1 2353 1 2354 1 2355 1 2356 1 2357 1 2358 1 2359 1 2360 1 2361 1 2362 1 2363 1 2364 1 2365 1 2366 1 2367 1 2368 1 2369 1 2370 1 2371 1 2372 1 2373 1 2374 1 2375 1 2376 1 2377 1 2378 1 2379 1 2380 1 2381 1 2382 1 2383 1 2384 1 2385 1 2386 1 2387 1 2388 1 2389 1 2390 1 2391 1 2392 1 2393 1 2394 1 2395 1 2396 1 2397 1 2398 1 2399 1 2400 1 2401 1 2402 1 2403 1 2404 1 2405 1 2406 1 2407 1 2408 1 2409 1 2410 1 2411 1 2412 1 2413 1 2414 1 2415 1 2416 1 2417 1 2418 1 2419 1 2420 1 2421 1 2422 1 2423 1 2424 1 2425 1 2426 1 2427 1 2428 1 2429 1 2430 1 2431 1 2432 1 2433 1 2434 1 2435 1 2436 1 2437 1 2438 1 2439 1 2440 1 2441 1 2442 1 2443 1 2444 1 2445 1 2446 1 2447 1 2448 1 2449 1 2450 1 2451 1 2452 1 2453 1 2454 1 2455 1 2456 1 2457 1 2458 1 2459 1 2460 1 2461 1 2462 1 2463 1 2464 1 2465 1 2466 1 2467 1 2468 1 2469 1 2470 1 2471 1 2472 1 2473 1 2474 1 2475 1 2476 1 2477 1 2478 1 2479 1 2480 1 2481 1 2482 1 2483 1 2484 1 2485 1 2486 1 2487 1 2488 1 2489 1 2490 1 2491 1 2492 1 2493 1 2494 1 2495 1 2496 1 2497 1 2498 1 2499 1 2500 1 2501 1 2502 1 2503 1 2504 1 2505 1 2506 1 2507 1 2508 1 2509 1 2510 1 2511 1 2512 1 2513 1 2514 1 2515 1 2516 1 2517 1 2518 1 2519 1 2520 1 2521 1 2522 1 2523 1 2524 1 2525 1 2526 1 2527 1 2528 1 2529 1 2530 1 2531 1 2532 1 2533 1 2534 1 2535 1 2536 1 2537 1 2538 1 2539 1 2540 1 2541 1 2542 1 2543 1 2544 1 2545 1 2546 1 2547 1 2548 1 2549 1 2550 1 2551 1 2552 1 2553 1 2554 1 2555 1 2556 1 2557 1 2558 1 2559 1 2560 1 2561 1 2562 1 2563 1 2564 1 2565 1 2566 1 2567 1 2568 1 2569 1 2570 1 2571 1 2572 1 2573 1 2574 1 2575 1 2576 1 2577 1 2578 1 2579 1 2580 1 2581 1 2582 1 2583 1 2584 1 2585 1 2586 1 2587 1 2588 1 2589 1 2590 1 2591 1 2592 1 2593 1 2594 1 2595 1 2596 1 2597 1 2598 1 2599 1 2600 1 2601 1 2602 1 2603 1 2604 1 2605 1 2606 1 2607 1 2608 1 2609 1 2610 1 2611 1 2612 1 2613 1 2614 1 2615 1 2616 1 2617 1 2618 1 2619 1 2620 1 2621 1 2622 1 2623 1 2624 1 2625 1 2626 1 2627 1 2628 1 2629 1 2630 1 2631 1 2632 1 2633 1 2634 1 2635 1 2636 1 2637 1 2638 1 2639 1 2640 1 2641 1 2642 1 2643 1 2644 1 2645 1 2646 1 2647 1 2648 1 2649 1 2650 1 2651 1 2652 1 2653 1 2654 1 2655 1 2656 1 2657 1 2658 1 2659 1 2660 1 2661 1 2662 1 2663 1 2664 1 2665 1 2666 1 2667 1 2668 1 2669 1 2670 1 2671 1 2672 1 2673 1 2674 1 2675 1 2676 1 2677 1 2678 1 2679 1 2680 1 2681 1 2682 1 2683 1 2684 1 2685 1 2686 1 2687 1 2688 1 2689 1 2690 1 2691 1 2692 1 2693 1 2694 1 2695 1 2696 1 2697 1 2698 1 2699 1 2700 1 2701 1 2702 1 2703 1 2704 1 2705 1 2706 1 2707 1 2708 1 2709 1 2710 1 2711 1 2712 1 2713 1 2714 1 2715 1 2716 1 2717 1 2718 1 2719 1 2720 1 2721 1 2722 1 2723 1 2724 1 2725 1 2726 1 2727 1 2728 1 2729 1 2730 1 2731 1 2732 1 2733 1 2734 1 2735 1 2736 1 2737 1 2738 1 2739 1 2740 1 2741 1 2742 1 2743 1 2744 1 2745 1 2746 1 2747 1 2748 1 2749 1 2750 1 2751 1 2752 1 2753 1 2754 1 2755 1 2756 1 2757 1 2758 1 2759 1 2760 1 2761 1 2762 1 2763 1 2764 1 2765 1 2766 1 2767 1 2768 1 2769 1 2770 1 2771 1 2772 1 2773 1 2774 1 2775 1 2776 1 2777 1 2778 1 2779 1 2780 1 2781 1 2782 1 2783 1 2784 1 2785 1 2786 1 2787 1 2788 1 2789 1 2790 1 2791 1 2792 1 2793 1 2794 1 2795 1 2796 1 2797 1 2798 1 2799 1 2800 1 2801 1 2802 1 2803 1 2804 1 2805 1 2806 </pre>				


```

1457 3         if (
1458 4             {
1459 5                 // If we already have a current backup node,
1460 4                 // clear all marks
1461 4                 if (red->currentBackupNode)
1462 5                     { red->currentBackupNode->numMarkNodes (TRUE, FALSE);
1463 5                     }
1464 5                 if (NULL != red->currentBackupNode)
1465 5                     {
1466 5                         red->currentBackupNode->unlockWorkItems (context);
1467 5                     }
1468 4                 BackupNode *bnode = (BackupNode *) 0;
1469 4                 red->currentBackupNode->unlockWorkItems (context);
1470 4                 if (red->currentBackupNode->lockWorkItems (context))
1471 4                     {
1472 5                         return EP_RA_RECOVER_ML_LOCKED;
1473 5                     }
1474 5                 bnode->populate (context, POPULATE_CHILDREN);
1475 4                 // We found the one to backup
1476 4                 return E_SUCCESS;
1477 4             }
1478 3         }
1479 2     }
1480 1 }
1481 1
1482 1     return EP_RA_RECOVER_NONE_EXISTS;
1483 1 }

```

```

1467 //

```

Thu Jan 03 12:52:58 2008	RSTPL_SetNextBackup	Page 65 of 92	Thu Jan 03 12:52:58 2008	RSTPL_SetNextBackup	Page 66 of 92
1488	/.....		1547	2	
1489	* ..		1548	2	{
1490	* Set Next Backup		1549	3	// If we already have a current backup node, clear all marks
1491	* ..		1550	3	{
1492	* <i>Function Description:</i>				rctd->currentBackupNode->unmarkNode (TRUE, FALSE);
1493	* This routine must set the restore environment to the the next backup				if (NULL!=rctd->currentBackupNode)
1494	* of the current top level object.		1552	3	{
1495	* <i>Parameters:</i>		1553	3	rctd->currentBackupNode->unlockWorkItems (context);
1496	* context (i) - Pointer to the restore context		1554	3	}
1497	* ..		1555	3	rctd->currentBackupNode->lockWorkItems (context);
1498	* ..		1556	3	BackupNode *bnode= (BackupNode *)lastprev;
1499	* ..		1557	3	if (!rctd->currentBackupNode->lockWorkItems (context))
1500	* <i>Return Codes:</i>		1558	3	{
1501	* E_SUCCESS - operation completed successfully		1559	3	return EP_RB_RECOVER_WT_LOCKED;
1502	* EP_RB_RECOVER_NO_MGMT_CATALOG - when at the most recent		1560	3	}
	EP_RB_RECOVER_PERMISSION_DENIED - when user cannot access file		1561	3	return EP_RB_RECOVER_WT_UNLOCKED;
	of the new catalog		1562	3	bnode->populateContext (PRIVATE_CHILDREN);
	EP_RB_RECOVER_NO_CATALOG - when mount_set_mcpname failed		1563	3	// We found the one to backup
	*****		1564	3	return E_SUCCESS;
1503			1565	2	}
1504			1566	1	return EP_RB_RECOVER_NONE_EXISTS;
1505			1567	1	}
1506			1570	1	
1507			1571		
1508	errno_t RSTPL_SetNextBackup (restore_context *context,				
1509	RSTRPC_backup_flags_t flags_t)				
1510	{				
1511	struct restoreContextData *rctd= (struct restoreContextData *){				
1512	context->appData;				
1513	}				
1514					
1515	WSetNode *wlist=rctd->currentWsetNode;				
1516	if (!wlist)				
1517	{				
1518	return EP_RB_RECOVER_RC_APP_DATA_NULL;				
1519	}				
1520					
1521					
1522					
1523	BackupNode *bnode=rctd->currentBackupNode;				
1524	if (!bnode)				
1525	{				
1526	return EP_RB_RECOVER_RC_APP_DATA_NULL;				
1527	}				
1528					
1529					
1530					
1531	RestoreNode *lastprev=NULL;				
1532					
1533	RestoreNode *rnode=wlist->getFirstChildId();				
1534	while (rnode && rnode != bnode)				
1535	{				
1536	if (flags==BACKUP_SELECTION_FLAG_PARTIAL_OK rnode->getRestoreable(
1537	STRATE_COMPLETED)				
1538	{				
1539	lastprev=rnode;				
1540	}				
1541	rnode=wlist->getNextChildId();				
1542	}				
1543					
1544					
1545	if (!lastprev)				

1578

//

```

1579
1580
1581 *
1582 * Set Most Recent Backup
1583 *
1584 * Function Description:
1585 *   Set the restore context to that of the most recent backup
1586 *   catalog.
1587 *
1588 * Parameters:
1589 *   context (I) - Pointer to the restore context
1590 *   flags (I) - Backup constraint flags: e.g., full-only/partial-ok
1591 *
1592 * Return Codes:
1593 *   R_SUCCESS - operation completed successfully
1594 *   EP_RB_RECOVER_PERMISSION_DENIED - when user cannot access the
1595 *   file of
1596 *   the new catalog.
1597
1598 *****
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
39
```

```
1636 3     }
1637 3     BackupNode *bnode= (BackupNode *)rn;
1638 3     rcd->currentBackupNode=bnode;
1639 3     if(!rcd->currentBackupNode->lockForItems(context))
1640 4     {
1641 4         return EP_RB_RECOVER_ML_LOCKED;
1642 3     }
1643 3     bnode->openLeaf(context.PORTRATE_CHILDREN);
1644 3     // We found the one to backup
1645 3     return E_SUCCESS;
1646 3     }
1647 2     }
1648 1     }
1651 1     return EP_RB_RECOVER_SPT_BUTIME_ERROR;
1652 }
```

```
1657 //
```



```
1789 3      (
1790 3      )
1791 2      else
1792 2      {
1793 3      {
1794 3      {
1795 4      {
1796 4      {
1797 4      {
1798 4      {
1799 2      }
1800 1      }
1801 1      }
1802 1      return E_SUCCESS;
1803 1      )
}
```

```
1807      //
```

```

1808 .....
1809 * Is There Next Backup
1810 .....
1811 * Function Description:
1812 * Determine if a backup exists after the backup that is
1813 * currently selected.
1814 .....
1815 * Parameters:
1816 * context (t) - Pointer to the restore context
1817 *
1818 * If (0) - TRUE/FALSE flags: e.g., full-only/partial-ok
1819 * isthere (0) - TRUE/FALSE that requested backup does exist
1820 .....
1821 * Return Codes:
1822 * E_SUCCESS - operation completed successfully
1823 * E_PL_RECOVER_xxx - when errors occur accessing catalog
1824 .....
1825 .....
1826 .....
1827 .....
1828 .....
1829 {
1830     struct restoreContextData *rcd(struct restoreContextData *) (
1831         context->appData);
1832     .....
1833     *isthere=FALSE;
1834     .....
1835     WtSetNode *wlist=rcd->currentWtSetNode;
1836     .....
1837     if(wlist)
1838     {
1839         return EP_RB_RECOVER_RC_APP_DATA_NULL;
1840     }
1841     .....
1842     BackupNode *bnode=rcd->currentBackupNode;
1843     .....
1844     if(!bnode)
1845     {
1846         return EP_RB_RECOVER_RC_APP_DATA_NULL;
1847     }
1848     .....
1849     RestoreNode *rn=wlist->getFirstChild();
1850     while(rn && rn != bnode)
1851     {
1852         if (flags==BACKUP_SELECTION_FLAG_PARTIAL_OK || rn->getSelectable(
1853             STATE_COMPLETE))
1854         {
1855             *isthere=TRUE;
1856             return E_SUCCESS;
1857         }
1858         rn=wlist->getNextChild();
1859     }
1860     return E_SUCCESS;
1861 }
1862 .....
1863 .....
1864 .....
1865 .....
1866 .....
1867 .....
1868 .....
1869 .....
1870 .....
1871 .....
1872 .....
1873 .....
1874 .....
1875 .....
1876 .....
1877 .....
1878 .....
1879 .....
1880 .....
1881 .....
1882 .....
1883 .....
1884 .....
1885 .....
1886 .....
1887 .....
1888 .....
1889 .....
1890 .....
1891 .....
1892 .....
1893 .....
1894 .....
1895 .....
1896 .....
1897 .....
1898 .....
1899 .....
1900 .....
1901 .....
1902 .....
1903 .....
1904 .....
1905 .....
1906 .....
1907 .....
1908 .....
1909 .....
1910 .....
1911 .....
1912 .....
1913 .....
1914 .....
1915 .....
1916 .....
1917 .....
1918 .....
1919 .....
1920 .....
1921 .....
1922 .....
1923 .....
1924 .....
1925 .....
1926 .....
1927 .....
1928 .....
1929 .....
1930 .....
1931 .....
1932 .....
1933 .....
1934 .....
1935 .....
1936 .....
1937 .....
1938 .....
1939 .....
1940 .....
1941 .....
1942 .....
1943 .....
1944 .....
1945 .....
1946 .....
1947 .....
1948 .....
1949 .....
1950 .....
1951 .....
1952 .....
1953 .....
1954 .....
1955 .....
1956 .....
1957 .....
1958 .....
1959 .....
1960 .....
1961 .....
1962 .....
1963 .....
1964 .....
1965 .....
1966 .....
1967 .....
1968 .....
1969 .....
1970 .....
1971 .....
1972 .....
1973 .....
1974 .....
1975 .....
1976 .....
1977 .....
1978 .....
1979 .....
1980 .....
1981 .....
1982 .....
1983 .....
1984 .....
1985 .....
1986 .....
1987 .....
1988 .....
1989 .....
1990 .....
1991 .....
1992 .....
1993 .....
1994 .....
1995 .....
1996 .....
1997 .....
1998 .....
1999 .....
2000 .....

```

```

1898 //*****
1899 * Get Top Level Templates:
1900 *
1901 * This function is required to retrieve the templates with which a
1902 * object could have been backed up.
1903 *
1904 * Parameters:
1905 *   (1) - pointer to the restore context
1906 *   (2) - the top level object
1907 *   templates - pointer to receive the start of the list of templates
1908 *   numberEntries - the real number of templates returned in the list
1909 *
1910 *****
1911 GetTopLevelTemplates(
1912     restoreContext_t *context,
1913     struct RSTRPC_top_level_obj *topLevelObj,
1914     struct RSTRPC_name_list **templates,
1915     int *numberEntries)
1916 {
1917     char *c1=(char *) (topLevelObj->appData.data);
1918     RestoreNode *nodeapp=(RestoreNode *) (c1);
1919     if (NULL==nodeapp)
1920     {
1921         the_log_stats(
1922             0, "RSTPL_GetTopLevelTemplates: Mark object has no app data");
1923         return BP_RA_RECOVER_RA_APP_DATA_NULL;
1924     }
1925     // Since we are setting a top level object, the marks on any existing
1926     // object must be cleared.
1927     //
1928     // Structure restoreContextData *rtd=(struct restoreContextData *) (
1929     //     context->appData);
1930     if (NULL!=rtd)
1931     {
1932         if (NULL!=rtd->currentBackupNode)
1933         {
1934             rtd->currentBackupNode->unmarkNode(TRUE, FALSE);
1935         }
1936     }
1937     //
1938     *numberEntries=NULL;
1939     *templates=NULL;
1940     //
1941     for (RestoreNode *rn=nodeapp->getFirstChild();
1942          rn;
1943          rn=nodeapp->getNextChild())
1944     {
1945         if (rn->nodeType() != RSTRPC_BACKUP)
1946         {
1947             continue;
1948         }
1949     }
1950 }
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013

```

```

1994 2 //*****
1995 3 * Get Top Level Templates:
1996 4 *
1997 5 * This function is required to retrieve the templates with which a
1998 6 * object could have been backed up.
1999 7 *
2000 8 * Parameters:
2001 9 *   (1) - pointer to the restore context
2002 10 *   (2) - the top level object
2003 11 *   templates - pointer to receive the start of the list of templates
2004 12 *   numberEntries - the real number of templates returned in the list
2005 13 *
2006 14 *****
2007 15 GetTopLevelTemplates(
2008 16     restoreContext_t *context,
2009 17     struct RSTRPC_top_level_obj *topLevelObj,
2010 18     struct RSTRPC_name_list **templates,
2011 19     int *numberEntries)
2012 20 {
2013 21     char *c1=(char *) (topLevelObj->appData.data);
2014 22     RestoreNode *nodeapp=(RestoreNode *) (c1);
2015 23     if (NULL==nodeapp)
2016 24     {
2017 25         the_log_stats(
2018 26             0, "RSTPL_GetTopLevelTemplates: Mark object has no app data");
2019 27         return BP_RA_RECOVER_RA_APP_DATA_NULL;
2020 28     }
2021 29     // Since we are setting a top level object, the marks on any existing
2022 30     // object must be cleared.
2023 31     //
2024 32     // Structure restoreContextData *rtd=(struct restoreContextData *) (
2025 33     //     context->appData);
2026 34     if (NULL!=rtd)
2027 35     {
2028 36         if (NULL!=rtd->currentBackupNode)
2029 37         {
2030 38             rtd->currentBackupNode->unmarkNode(TRUE, FALSE);
2031 39         }
2032 40     }
2033 41     *numberEntries=NULL;
2034 42     *templates=NULL;
2035 43     //
2036 44     for (RestoreNode *rn=nodeapp->getFirstChild();
2037          rn;
2038          rn=nodeapp->getNextChild())
2039     {
2040         if (rn->nodeType() != RSTRPC_BACKUP)
2041         {
2042             continue;
2043         }
2044     }
2045 }
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4100
4101
4102
4103
4104
4105
4106
4107

```

Thu Jan 03 12:52:58 2008	RSTPL_GetTopLevelTemplates	Page 81 of 92	Thu Jan 03 12:52:58 2008	RSTPL_ClearRestoreContext	Page 82 of 92
2013 3	if(NULL == entity->name)		2011	/*	
2014 4	{		2012	Clear Restore Context :	*****
2015 4	rbe_log_fatal(2013	return E_SUCCESS;	
2016 4	0, "RSTPL_GetTopLevelTemplates: Malloc failure");		2014	This function is called to notify the plug-in that its currently	selected
2017 3	return EP_RB_RECOVER_MALLOCFailure;			top level object is no longer selected	
2018 3	}			plug-in should perform whatever cleanup and memory deallocation is	appropriate.
2019 3	*templateas=entity;		2019	Returns:	
2020 3	(*numberofentries)++;		2020	E_SUCCESS	on success
2021 3	}		2021	EP_RB_RECOVER_XXX	on error
2022 2			2022	* context	
2023 1	}		2023	* context	(I) - Pointer to the restore context
2024 1			2024	* context	
2025 1	return E_SUCCESS;		2025	* context	
2026 1	}		2026	* context	
2027 1	return E_SUCCESS;		2027	* context	
2028 1	}		2028	* context	
			2029	* context	
			2030	* context	
			2031	* context	
			2032	* context	
			2033	* context	
			2034	* context	
			2035	* context	
			2036	* context	
			2037	* context	
			2038	* context	
			2039	* context	
			2040	* context	
			2041	* context	
			2042	* context	
			2043	* context	
			2044	* context	
			2045	* context	
			2046	* context	
			2047	* context	
			2048	* context	
			2049	* context	
			2050	* context	
			2051	* context	
			2052	* context	
			2053	* context	
			2054	* context	
			2055	* context	
			2056	* context	
			2057	* context	
			2058	* context	
			2059	* context	
			2060	* context	
			2061	* context	
			2062	* context	
			2063	* context	
			2064	* context	
			2065	* context	
			2066	* context	
			2067	* context	
			2068	* context	
			2069	* context	
			2070	* context	
			2071	* context	
			2072	* context	

```

2075 /*****
2076 * Is There Prev Backup For Time
2077 *
2078 * Function Description:
2079 * Determine if a backup exists prior to the specified time
2080 *
2081 * Parameters:
2082 * context (I) - Pointer to the restore context
2083 * flags (I) - Time for the query
2084 *
2085 * Isthree (O) - TRUE/FALSE that requested backup does exist
2086 *
2087 * Return Codes:
2088 * E_SUCCESS - operation completed successfully
2089 * E_RL_RECOVER_CXX - when errors occur decreasing catalogs
2090 *
2091 *****/
2092
2093 errno_t RSTPL_IsTherePrevBackupForTime( restore_context *context,
2094                                         const time_t thistime,
2095                                         RSTPL_Backup_flags_t flags,
2096                                         boolean_t isthree )
2097 {
2098     *isthree=FALSE;
2099
2100     struct restoreContextData *rcd=(struct restoreContextData *){
2101         context->appdata);
2102
2103     WlSetNode *wliset=rcd->currentWlsetNode;
2104
2105     if(!wliset)
2106     {
2107         return EP_RL_RECOVER_RC_APP_DATA_NULL;
2108     }
2109
2110     BackupNode *bnode=rcd->currentBackupNode;
2111
2112     if(!bnode)
2113     {
2114         return EP_RL_RECOVER_RC_APP_DATA_NULL;
2115     }
2116
2117     // Loop over all backup objects in this set looking for one which is
2118     // previous
2119     // to this one.
2120     for(RestoreNode *rn=wliset->getNextChild();rn;
2121         rn=wliset->getNextChild())
2122     {
2123         if((flags==BACKUP_SELECTON_FLAG_PARTIAL_OK) || rn->getStatebit(
2124             STATE_COMPLETE))
2125         {
2126             BackupNode *bn=(BackupNode *)rn;
2127             RmPropertyDate *dps(RmPropertyDate *PROPERTY_BACKUP_DATE));
2128             time_t btime=dps->getValue()->mixtime());
2129         }
2130     }

```

```

2131     if(thistime > btime)
2132     {
2133         *isthree=TRUE;
2134         return E_SUCCESS;
2135     }
2136
2137     }
2138
2139     return E_SUCCESS;
2140 }

```



```
2280 /*****
2281  * Does Alternate Exist
2282  */
2283  * This routine determines if an alternate trailset exists for the
2284  * given template.
2285  *
2286  * Parameters:
2287  *   trailset (T) - Pointer to the restore context
2288  *   templateName (T) - The name of the template to look for
2289  *   exists
2290  *   0) - Return flag for whether or not the alternate exists
2291  *
2292  *****/
2293  restore_by
2294  RSTPL_DoesAlternateExist( restore_context *context,
2295  const template_name_t *templateName,
2296  boolean_t *exists )
2297  {
2298  /*****
2299  *
2300  *   TEMPORARY,
2301  *   until we get the real version: *****/
2302  *
2303  * exists = FALSE;
2304  * return E_SUCCESS;
2305  */
2306  }
```

